INICIACIÓN A LA ROBÓTICA Y PROGRAMACIÓN

Región de Murcia

Unión Europea

Fondo Europeo de Desarrollo Regional - Una manera de hacer Europa

ÍNDICE

1. Lista de materiales y presentación

2. Introducción a Arduino

3. Instalación

4. Introducción a la programación

    4.1. Scratch

    4.2. Arduino IDE

5. Prácticas

    5.1. Práctica de iniciación: Luz intermitente
    5.2. Aviso para invidentes
    5.3. Semáforo
    5.4. Semáforo con aviso para invidentes
    5.5. Semáforo con pulsador
    5.6. Cruce de semáforos
    5.7. Encendido nocturno (LDR)
    5.8. Cadena de montaje
    5.9. Ventilador automático
    5.10. Luz roja, verde y azul (RGB)
    5.11. Arco iris de luz (RGB)
    5.12. Detector de objetos (CNY70)
    5.13. Movimiento de un servomotor 180º
    5.14. Movimiento de un servomotor con un potenciómetro
    5.15. Medidor de distancias (ULTRASÓNICO)
    5.16. Conexión de Display LCD
    5.17. Control de leds con mando a distancia
    5.18. Motor paso a paso

6. Panel de control Picuino

    6.1 Introducción
    6.2 ¿Por qué utilizar picuino?
    6.3 Instalación
    6.4 Conexión de picuino con arduino
    6.5 Instrucciones básicas propias de picuino
    6.6 Control picuino por bluetooth (práctica 19)

Todas las prácticas están resueltas en el archivo www.microlog.net/robotica_murcia.zip

Enlace Arduino Microlog: www.microlog.es/entrenador_arduino.zip

1 Maletín transparente con asa y doble cierre **LOG 5051**

- Medidas: 240x320x60 mm

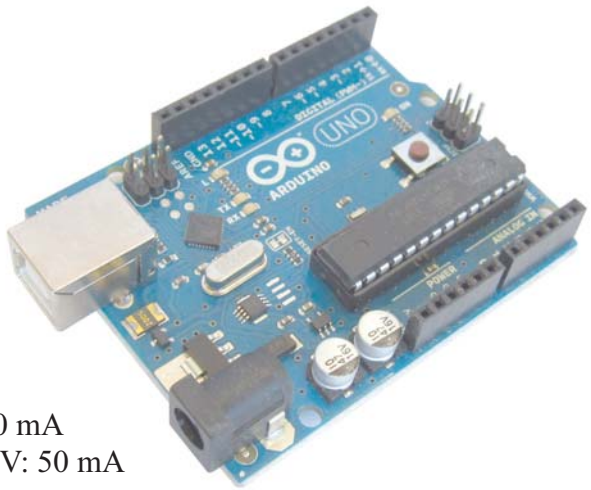
1 Estuche de 160x30x122 mm **LOG 5050**

- 7 compartimentos
- Plástico transparente

PARTIDA 4.2.1. PLACA CONTROLADORA O UNIDAD CENTRAL

1 Placa Arduino UNO, funciona de manera autónoma **LOG 4031**

- Microprocesador: ATmega328
- Tensión operativa: 5V
- Tensión de alimentación: 7-12 V
- Entradas digitales: 14
- Salidas digitales: 14
- Entradas analógicas: 6
- Memoria flash: 32Kb
- SRAM: 2Kb
- EEPROM: 1Kb
- Velocidad del reloj: 16MHz
- Máxima corriente continua para entradas: 40 mA
- Máxima corriente continua para los pins 3,3V: 50 mA



PARTIDA 4.2.2. CABLE USB DE CONEXIÓN ENTRE LA PLACA CONTROLADORA Y ORDENADOR

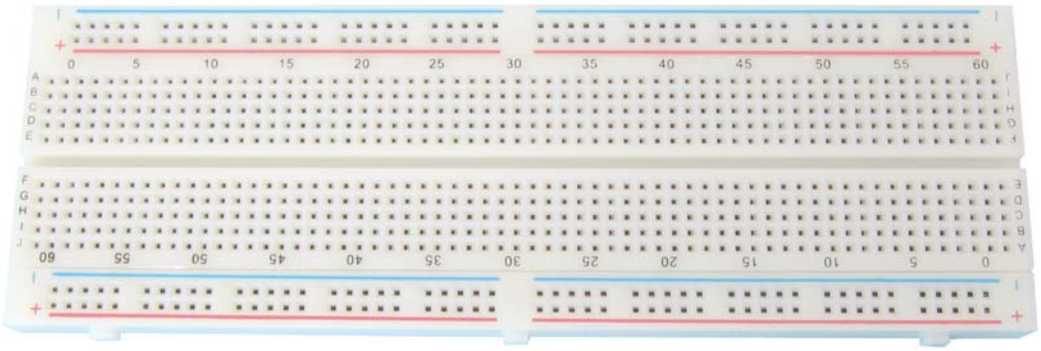
1 Cable USB A-B 1,8 metros **LOG 4009**

- 1,8 metros

PARTIDA 4.2.3. PLACA PROTOBOARD

1 Placa protoboard de 830 puntos **LOG 887**

- 830 contactos
- Serigrafía de columnas, filas y polaridad



PARTIDA 4.2.4. CABLE DE CONEXIÓN

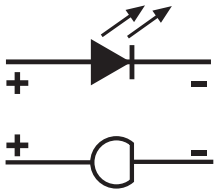
50 Latiguillos para placa protoboard macho-macho **LOG 9519G**

PARTIDA 4.2.5. DIODOS LED 5 mm

3 Diodos LED verde de 5 mm **LOG 722**

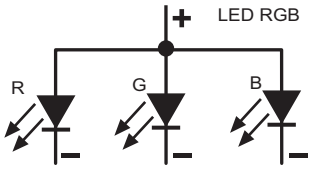
3 Diodos LED ámbar de 5 mm **LOG 723**

3 Diodos LED rojo de 5 mm **LOG 724**



3 Diodos LED blanco de 5 mm alta luminosidad **LOG 720**

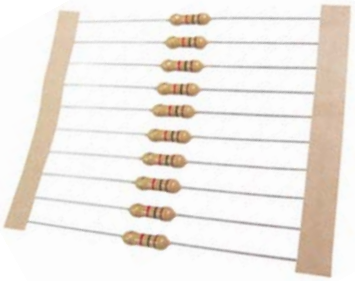
3 Diodos LED RGB de 5 mm. ánodo común **LOG 730**



PARTIDA 4.2.6. RESISTENCIAS

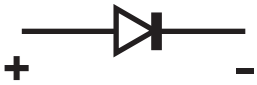
10 Resistencias de 220 Ω 1/2 W 5 % **LOG 748X220**
(rojo-rojo-marrón)
10 Resistencias de 330 Ω 1/2 W 5 % **LOG 748X330**
(naranja-naranja-marrón)
10 Resistencias de 100 Ω 1/2 W 5 % **LOG 748X100**
(marrón-negro-marrón)
10 Resistencias de 1KΩ 1/2 W 5 % **LOG 748X1K**
(marrón-negro-rojo)
10 Resistencias 4,7KΩ 1/2 W 5 % **LOG 748X4K7**
(amarillo-morado-rojo)
10 Resistencias de 10KΩ 1/2 W 5 % **LOG 748X10K**
(marrón-negro-naranja)

Símbolo utilizado



PARTIDA 4.2.7. DIODOS

10 Diodos 1N4007 **LOG 760X**



PARTIDA 4.2.8. ZUMBADORES

2 Zumbadores piezo-eléctrico activos **LOG 7714**

PARTIDA 4.2.9. MICROINTERRUPTORES

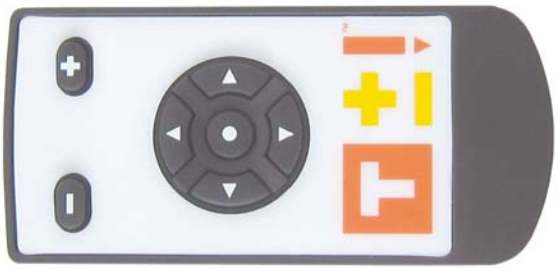
5 Interruptores palanca para placas board y C.I. **LOG 529**

PARTIDA 4.2.10. EMISOR - RECEPTOR DE INFRARROJOS

1 Sensor TSOP 4838 **LOG 835**

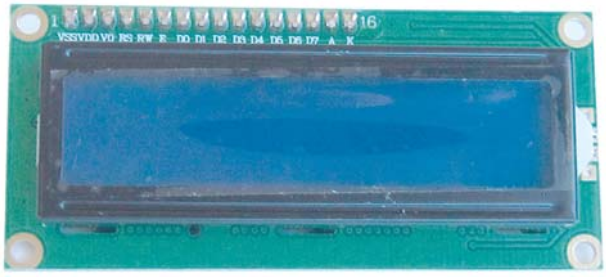
1 Mando a distancia **LOG 4000**

- 7 Botones:
- 1 Disminuir
- 1 Incrementar
- 4 Flechas de dirección
- 1 Central OK



PARTIDA 4.2.11. PANTALLA LCD 1602

1 Display LCD, muestra mensajes rápidamente.
Librerías y drivers especificos. **LOG 4069**

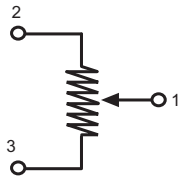


PARTIDA 4.2.12. FOTORESISTENCIAS LDR

5 LDR (Resistencia sensible a la luz) **LOG 731**

PARTIDA 4.2.13. POTENCIÓMETROS

2 Potenciómetros de 10K ohmios, 5 %, **LOG 742**



PARTIDA 4.2.14. SERVO MOTORES

2 Miniservo **LOG 06**

- Dimensiones 23x12,2x29 mm
- Rotación 180º
- Analógico
- Torque estático: 4,5 V > 1, 5 kg·cm

PARTIDA 4.2.15. MOTOR CON POLEA

1 Motor de corriente contínua **LOG 02**

- 1,5 a 6 V
- Diámetro 24 mm
- Longitud 29 mm
- Eje de 2 mm
- 9.800 r.p.m. en vacio a 4,5 V
- Potencia > 0,5 W

1 Polea de plástico **LOG 222P**

PARTIDA 4.2.16. SENSORES DE TEMPERATURA

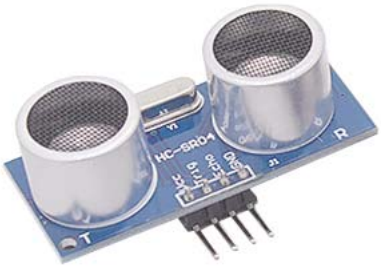
3 Sensores de temperatura LM35 **LOG 733**

PARTIDA 4.2.17. SENSOR DE TEMPERATURA Y HUMEDAD INTEGRADO

1 Sensor de temperatura y humedad **LOG 4052**

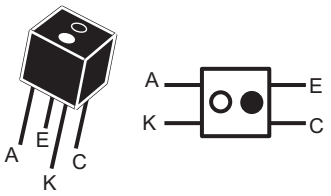
PARTIDA 4.2.18. SENSOR DE PROXIMIDAD

1 Sensores ultrasónico, permite medir
la distancia hasta un obstáculo de 2cm a 4m.
HC-SR04 **LOG 4046**



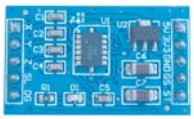
PARTIDA 4.2.19. SENSOR DE INFRARROJOS

2 Optoacopladores **CNY70 LOG 837**



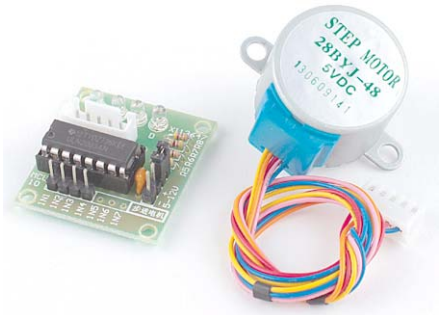
PARTIDA 4.2.20. SENSOR GIROSCOPIO TRES EJES

1 Sensor giroscopio tres ejes **LOG 4045**



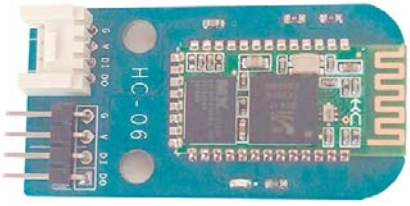
PARTIDA 4.2.21. MOTOR PASO A PASO MAS CONTROLADOR

1 Motor paso a paso y controlador **LOG 14P**



PARTIDA 4.2.22. MÓDEM BLUETOOTH

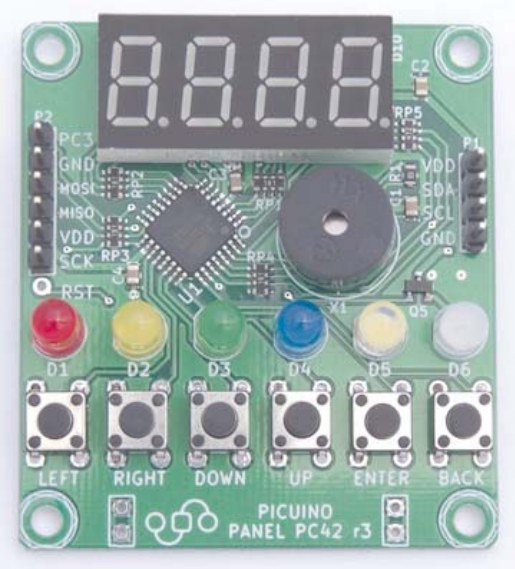
1 Bluetooth Shield HC-06. Permite la transmisión inalámbrica. **LOG 4058**



PARTIDA 4.2.23. PLACA AUXILIAR DE APENDIZAJE

1 **PICUINO LOG 4055**

- 6 pulsadores normalmente abierto
- 5 LEDs (Rojo, amarillo, verde, azul, blanco)
- 1 LED RGB
- 1 Zumbador
- 1 Display de 4 digitos



Plataforma online de programas y ejemplos prácticos con Arduino y el Shield de prácticas: www.picuino.com

PARTIDA 4.2.24. LIBRO DE PRÁCTICAS

1 Manual impreso de programación y robótica Microlog de 83 páginas (este mismo manual).

Indice
- Introducción
- Scratch
- Arduino IDE
- Prácticas
- Panel de control Picuino

2. PRESENTACIÓN Y ACABADO DE LOS COMPONENTES

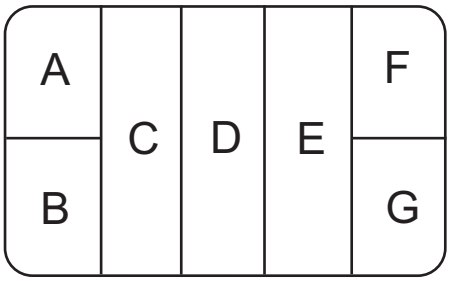
El kit básico se presenta en una maleta transparente que incluye en su interior un estuche con compartimentos, lo que facilita la localización de los componentes desde el exterior.
El estuche interior contiene 7 compartimentos, donde se aloja el pequeño material electrónico como resistencias, diodos led...
Cada artículo del lote viene etiquetado con el código de **MICROLOG**, su denominación para una rápida identificación y el número de partida.

CAJA KIT BÁSICO

PARTIDA	CANTIDAD	DESCRIPCIÓN
	1	Estuche Kit de componentes **LOG 5050**
4.2.1.	1	Placa controladora Arduino UNO **LOG 4031**
4.2.2.	1	Cable USB A-B **LOG 4009**
4.2.3.	1	Placa protoboard de 830 puntos **LOG 887**
4.2.4.	50	Latiguillos para placa protoboard macho-macho **LOG 9519G**
4.2.10.	1	Mando a distancia **LOG 4000** Sensor TSOP 4838 **LOG 835**
4.2.11.	1	Patalla LCD **LOG 4069**
4.2.14.	2	Miniservo **LOG 06**
4.2.17.	1	Sensor de temperatura y humedad **LOG 4052**
4.2.18.	1	Sensores ultrasónico (proximidad) HC-SR04 **LOG 4046**
4.2.20.	1	Sensor giroscopio tres ejes **LOG 4045**
4.2.21.	1	Motor paso a paso y controlador **LOG 14P**
4.2.22.	1	Bluetooth Shield HC-06. **LOG 4058**
4.2.23.	1	Placa auxiliar de aprendizaje **PICUINO LOG 4055**

ESTUCHE COMPONENTES ELECTRÓNICOS

PARTIDA	CANT.	DESCRIPCIÓN	COMPARTIMENTO
	1	Estuche de 160x30x122 mm con compartimentos **LOG 5050**	A
4.2.5.	3x5	Conjunto de diodos LED	D
4.2.6.	10x6	Conjunto de resistencias	C
4.2.7.	10	Diodos 1N4007 **LOG 760**	C
4.2.8.	2	Zumbadores piezo-eléctrico **LOG 7714**	G
4.2.9.	5	Interruptores palanca para placas board **LOG 529**	E
4.2.12.	5	LDR (Resistencia sensible a la luz) **LOG 731**	F
4.2.13.	2	Potenciómetro de 10K ohmios **LOG 742**	A
4.2.15.	1+1	Motor de corriente continua **LOG 02** Polea de plástico **LOG 222P**	B
4.2.16.	3	Sensores de temperatura LM35 **LOG 733**	F
4.2.19.	2	Optoacoplador **CNY70 LOG 837**	G



2. INTRODUCCIÓN A ARDUINO

Todo sistema robótico está formado por:

- Un sistema motriz, encargado de generar los movimientos del robot.

- Un conjunto de sensores, que ofrecerá al robot una visión del entorno que le rodea.

- Un sistema de control, que analizará los datos recibidos a través de los sensores y decidirá cómo debe actuar el robot en consecuencia, mostrando así cierta autonomía e inteligencia.

Para que el robot adquiera cierta inteligencia es muy importante saber elaborar la correcta programación del sistema de control.

Arduino UNO es una placa electrónica que podemos utilizar como sistema de control de nuestros robots gracias a su microcontrolador ATmega328. Contiene la siguiente distribución de pines:

- 14 pines digitales que pueden ser configurados como entradas o salidas, 6 de las cuales (serigrafiadas con el símbolo ~) pueden utilizarse tanto como señales digitales PWM o como salidas analógicas.

- 6 entradas analógicas serigrafiadas desde A0 hasta A5.

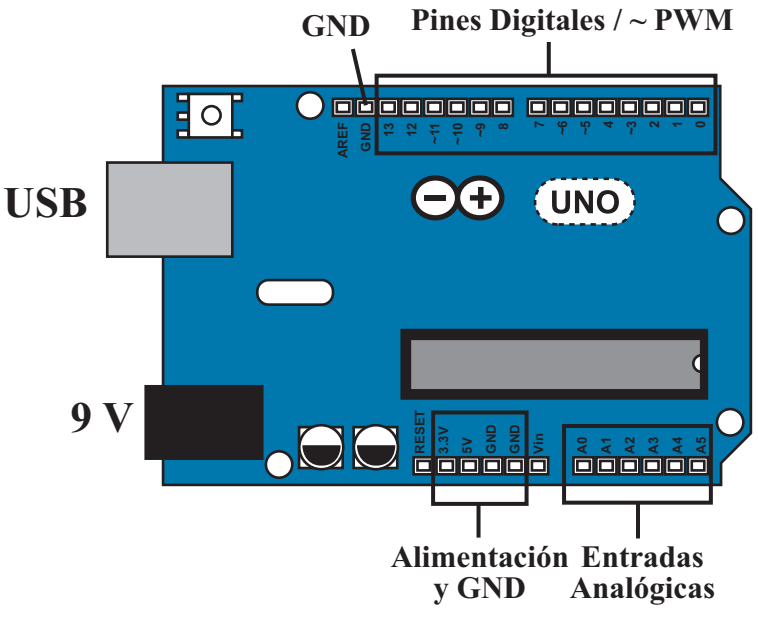
- 3 pines GND - toma de tierra.

- 1 Pin de alimentación a 5V.

- 1 Pin de alimentación a 3.3V.

- Conector de alimentación de la tarjeta a 9V.

- Conector de alimentación y programación puerto USB.



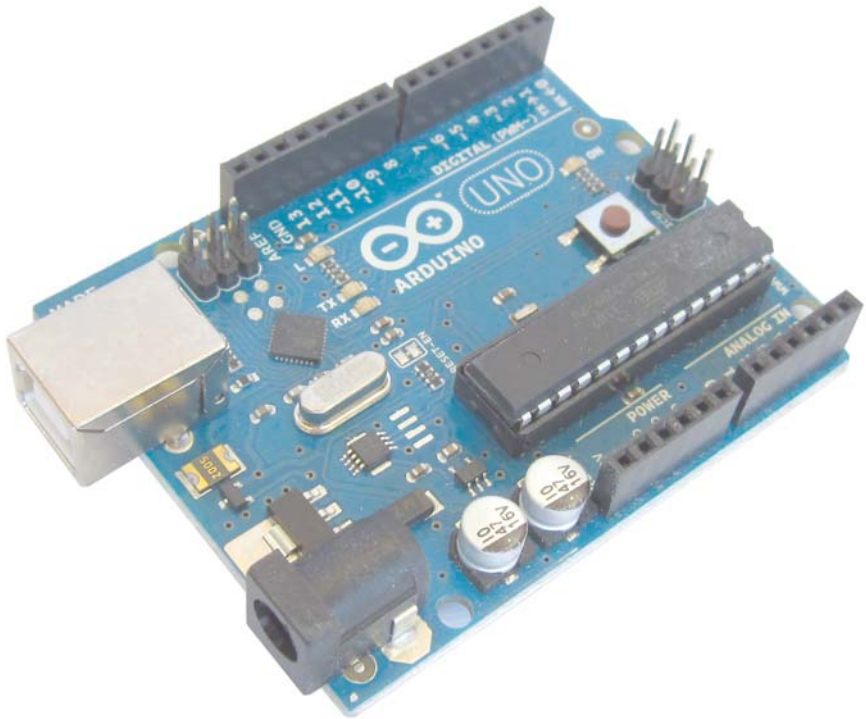
Podemos emplear diferentes leguajes para programar la placa Arduino UNO. Principalmente utilizaremos los siguientes lenguajes:

- Scratch: es un lenguaje muy utilizado en primaria y secundaria para acercar la programación a los niños de una forma muy intuitiva. Para trabajar con arduino empleamos S4A, una versión del Scratch adaptada a la placa. Este lenguaje podemos utilizarlo para realizar la mayoría de las prácticas sencillas pero pierde sentido en la elaboración de robots independientes (autónomos) porque la placa no tiene autonomía cuando se programa con S4A, precisando estar conectada siempre al PC.

- Ardublock: es un lenguaje basado en el uso de bloques, lo que facilita notablemente el proceso de desarrollo de los programas. La ventaja que ofrece frente a scratch (S4A) es que mantiene el carácter autónomo de la placa, siendo una buena alternativa para el diseño de robots.

- Arduino IDE (processing): es un lenguaje basado en líneas de código, cuya gramática supone cierta complejidad.

A lo largo de este manual, el alumno aprenderá a instalar y utilizar los diferentes sistemas de programación que puede emplear para programar su robot y posteriormente emplear aquel lenguaje que mejor se adapte a su nivel y proyecto.



3. INSTALACIÓN

Independientemente del lenguaje en el que vayamos a programar, Arduino IDE (processing) es indispensable que esté instalado en nuestro equipo.

INSTALANDO ARDUINO IDE

Primero conectamos la tarjeta controladora al ordenador a través del cable USB, y a continuación descargamos el software de programación desde la página www.arduino.org Dependiendo del sistema operativo que utilicemos procederemos de una de las siguientes maneras:

A. INSTALACIÓN EN LINUX

Nos hemos basado en la distribución Ubuntu para realizar esta guía de instalación puesto que la mayoría de las distribuciones de Linux a nivel educativo se basan en ella.

Para trabajar con arduino es preciso que nuestra distribución tenga instalados los siguientes paquetes además de Arduino IDE:

- sun's java runtime (jre)
- gcc-avr
- avr-libc
- binutils-avr

Para instalar estos paquetes se precisa la clave de administrador. Abrimos un terminal y ejecutamos las siguientes instrucciones:

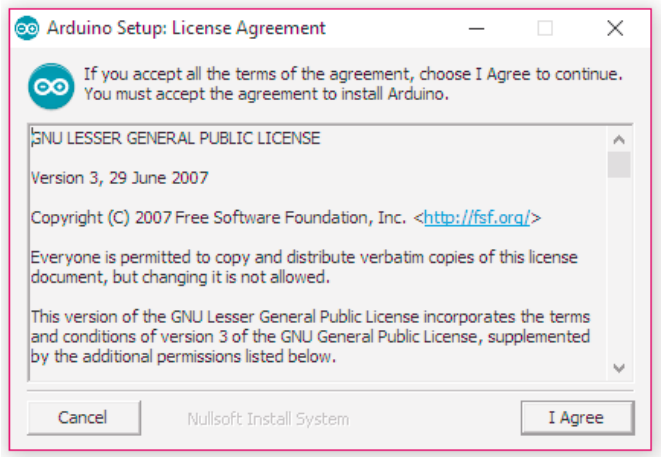
```
$ sudo add-apt-repository ppa:arduino-ubuntu-team
$ sudo apt-get update
$ sudo apt-get install arduino
```

Para ejecutar la aplicación Arduino IDE basta con acceder al menú Aplicaciones > Programación > Arduino

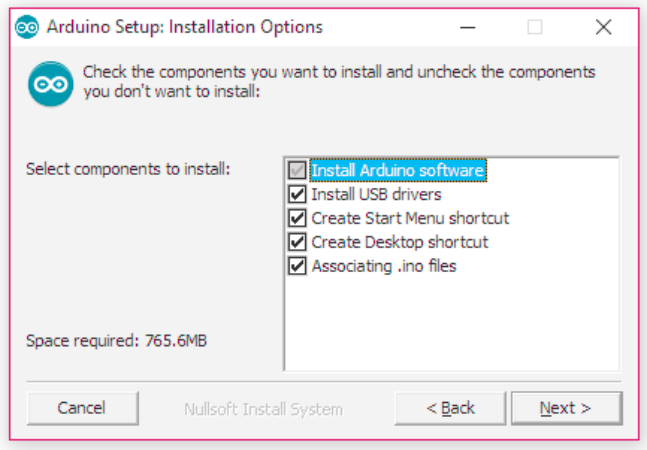
B. INSTALACIÓN EN WINDOWS

Descargamos el software de Arduino desde la sección "downloads" de www.arduino.org y procedemos a ejecutar el programa descargado.

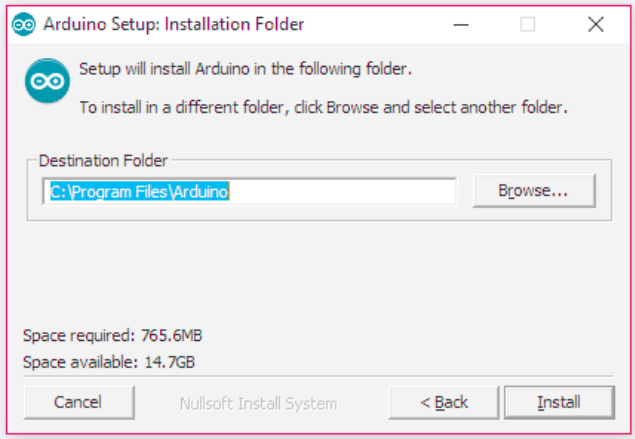
- Aceptamos el acuerdo de licencia pulsando “I Agree”


Arduino Setup: License Agreement
If you accept all the terms of the agreement, choose I Agree to continue.
You must accept the agreement to install Arduino.
GNU LESSER GENERAL PUBLIC LICENSE
Version 3, 29 June 2007
Copyright (C) 2007 Free Software Foundation, Inc. <http://fsf.org/>
Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.
This version of the GNU Lesser General Public License incorporates the terms and conditions of version 3 of the GNU General Public License, supplemented by the additional permissions listed below.
Cancel
Nullsoft Install System
I Agree

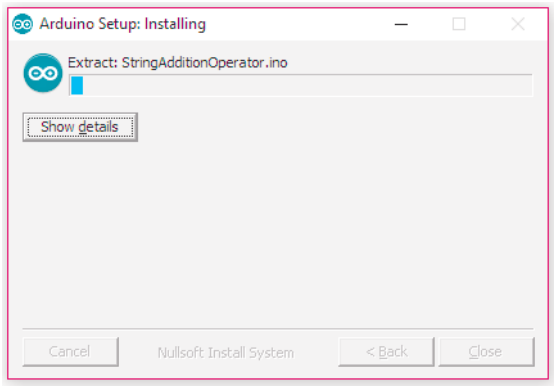
- Marcamos todos los componentes a instalar y pulsamos “next”


Arduino Setup: Installation Options
Check the components you want to install and uncheck the components you don't want to install:
Select components to install:
Install Arduino software
Install USB drivers
Create Start Menu shortcut
Create Desktop shortcut
Associating .ino files
Space required: 765.6MB
Cancel
Nullsoft Install System
< Back
Next >

- Seleccionamos la ruta donde se instalará el programa Arduino IDE y pulsamos “Install”



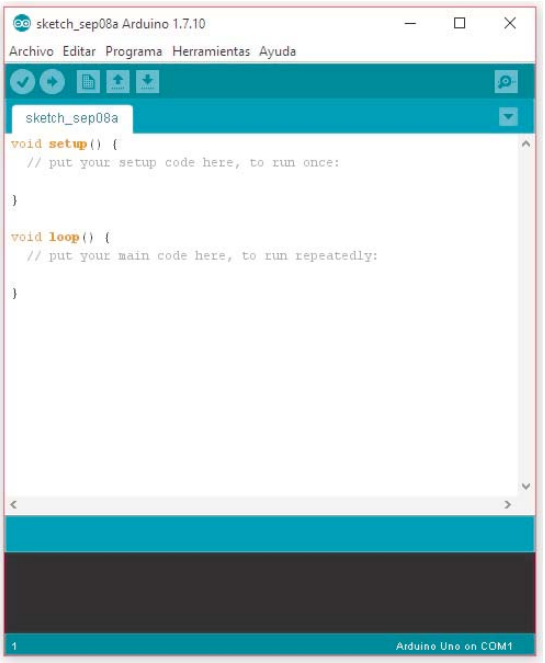
- Comienza la instalación



- Una vez finalizada la instalación pulsamos “Close”

En nuestro escritorio habrá el siguiente icono que dará acceso al sistema de programación Arduino IDE.

Al entrar en el programa veremos la siguiente pantalla.



En ella podemos crear nuestros propios programas introduciendo las líneas de código y posteriormente cargarlos en la tarjeta arduino utilizando la opción de menú *cargar* que se incluye en el menú *archivo*.

O bien podemos cargar directamente en la tarjeta arduino los ejemplos que podemos encontrar en la opción *ejemplos* del menú *archivos.*

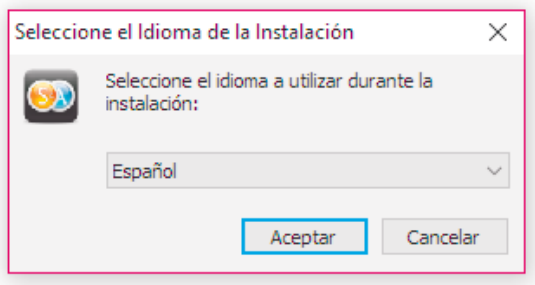
INSTALACIÓN DE SCRATCH (S4A)

Si el usuario se siente más cómodo con entornos de programación gráficos, es posible programar arduino con una variante de SCRATCH, esta versión se llama S4A.

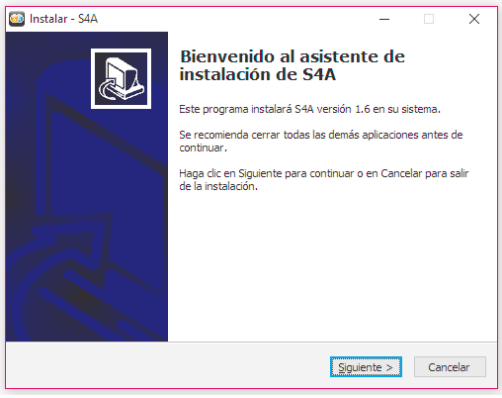
Para que funcione correctamente es preciso tener instalado el software Arduino IDE que hemos instalado en el capítulo anterior.

A continuación descargamos S4A desde el siguiente enlace: http://s4a.cat

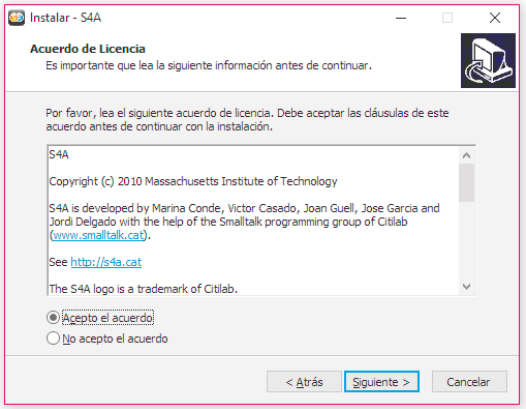
Ejecutamos el instalador descargado y seleccionamos el idioma de instalación.



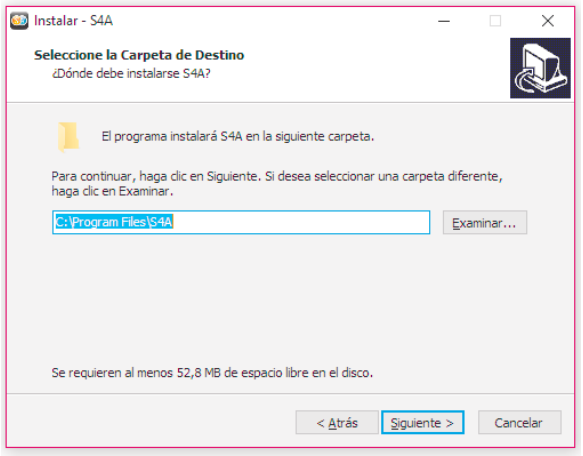
Pulsamos en siguiente.


Instalar - S4A
Bienvenido al asistente de instalación de S4A
Este programa instalará S4A versión 1.6 en su sistema.
Se recomienda cerrar todas las demás aplicaciones antes de continuar.
Haga clic en Siguiente para continuar o en Cancelar para salir de la instalación.
Siguiente >
Cancelar

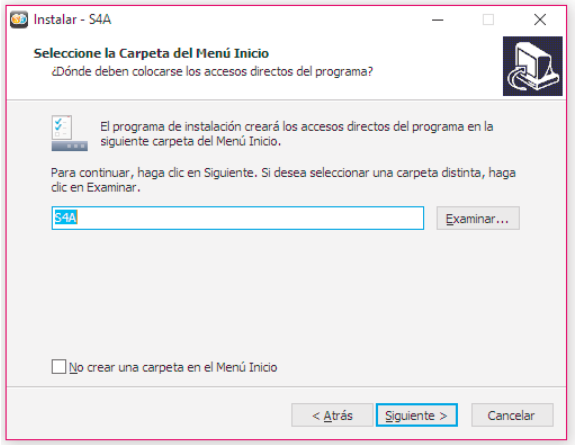
Aceptamos el acuerdo de licencia y pulsamos siguiente.


Instalar - S4A
Acuerdo de Licencia
Es importante que lea la siguiente información antes de continuar.
Por favor, lea el siguiente acuerdo de licencia. Debe aceptar las cláusulas de este acuerdo antes de continuar con la instalación.
S4A
Copyright (c) 2010 Massachusetts Institute of Technology
S4A is developed by Marina Conde, Victor Casado, Joan Guell, Jose Garcia and Jordi Delgado with the help of the Smalltalk programming group of Citilab (www.smalltalk.cat).
See http://s4a.cat
The S4A logo is a trademark of Citilab.
Acepto el acuerdo
No acepto el acuerdo
< Atrás
Siguiente >
Cancelar

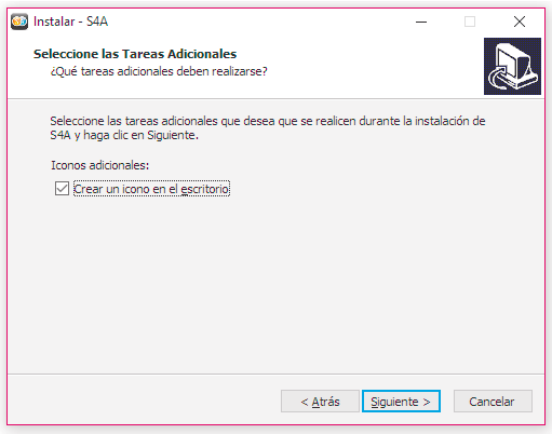
Seleccionamos la carpeta de destino.


Instalar - S4A
Seleccione la Carpeta de Destino
¿Dónde debe instalarse S4A?
El programa instalará S4A en la siguiente carpeta.
Para continuar, haga clic en Siguiente. Si desea seleccionar una carpeta diferente, haga clic en Examinar.
C:\Program Files\S4A
Examinar...
Se requieren al menos 52,8 MB de espacio libre en el disco.
< Atrás
Siguiente >
Cancelar

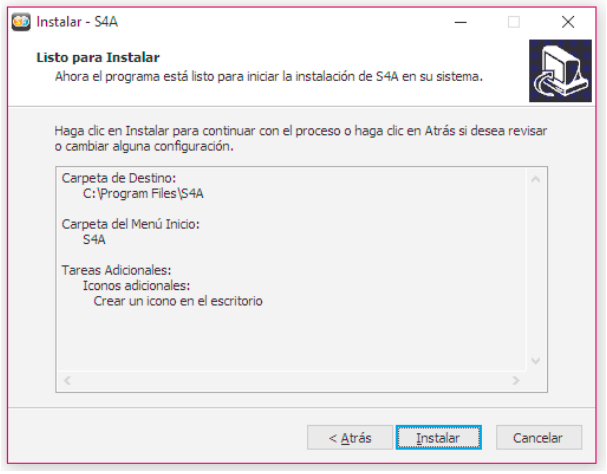
Seleccionamos el destino de los accesos directos.

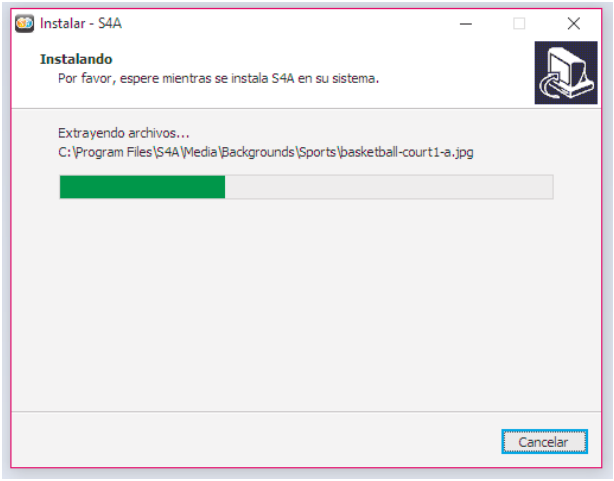

Instalar - S4A
Seleccione la Carpeta del Menú Inicio
¿Dónde deben colocarse los accesos directos del programa?
El programa de instalación creará los accesos directos del programa en la siguiente carpeta del Menú Inicio.
Para continuar, haga clic en Siguiente. Si desea seleccionar una carpeta distinta, haga clic en Examinar.
S4A
Examinar...
No crear una carpeta en el Menú Inicio
< Atrás
Siguiente >
Cancelar

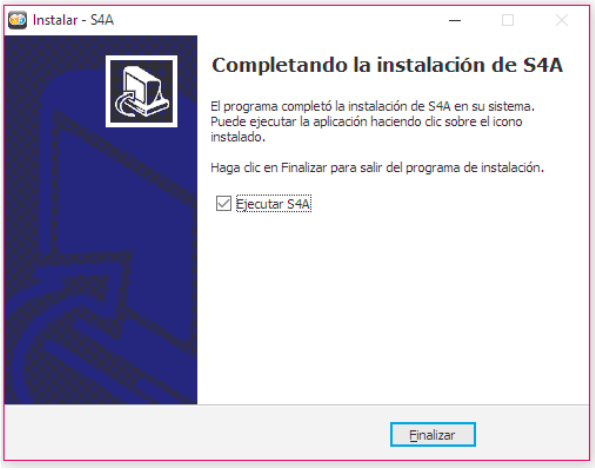
Indicamos si queremos crear un acceso directo en el escritorio.


Instalar - S4A
Seleccione las Tareas Adicionales
¿Qué tareas adicionales deben realizarse?
Seleccione las tareas adicionales que desea que se realicen durante la instalación de S4A y haga clic en Siguiente.
Iconos adicionales:
Crear un icono en el escritorio
< Atrás
Siguiente >
Cancelar

Y pulsamos instalar.


Instalar - S4A
Listo para Instalar
Ahora el programa está listo para iniciar la instalación de S4A en su sistema.
Haga clic en Instalar para continuar con el proceso o haga clic en Atrás si desea revisar o cambiar alguna configuración.
Carpeta de Destino:
C:\Program Files\S4A
Carpeta del Menú Inicio:
S4A
Tareas Adicionales:
Iconos adicionales:
Crear un icono en el escritorio
< Atrás
Instalar
Cancelar




Instalar - S4A
Completando la instalación de S4A
El programa completó la instalación de S4A en su sistema. Puede ejecutar la aplicación haciendo clic sobre el icono instalado.
Haga clic en Finalizar para salir del programa de instalación.
Ejecutar S4A
Finalizar

INSTALACIÓN DE LA TARJETA

Al instalar Arduino IDE, los drivers de la tarjeta quedan instalados en el PC. Para comprobar su correcto funcionamiento conectamos la tarjeta al ordenador a través del cable USB. Automáticamente el ordenador nos reconocerá la tarjeta.

Para introducir en ella los programas que hagamos, necesitamos saber qué puerto COM ha asignado el sistema operativo a la tarjeta:

- Pulsamos con el botón derecho en el icono de "equipo" en el panel de control de Windows.

- Pulsamos propiedades.

- Pulsamos administrador de dispositivos.

- Y desplegamos el menú "puertos com y lpt".

- En la lista desplegable veremos que uno de los puertos está asociado a la tarjeta Arduino UNO. Debemos recordar que número de puerto se ha asociado a la tarjeta para proceder a la carga de programas correctamente.

- Si la tarjeta Arduino UNO no aparece en el listado de puertos, el ordenador no está reconociendo correctamente la tarjeta.

PRIMEROS PASOS CON ARDUINO IDE

Para cargar un programa en la tarjeta desde Arduino IDE:

- Abrimos el programa.

- Seleccionamos del menú herramientas > puerto serial el puerto en el que se instaló la tarjeta Arduino UNO.

- Seleccionamos del menú herramientas > tarjeta el modelo de la tarjeta, en nuestro caso Arduino UNO.

- Seleccionamos un programa de ejemplo para probar la conectividad de la tarjeta. En el menú "archivo" seleccionamos ejemplos > 01. Basics > Blink.

- Seleccionamos del menú archivo la opción cargar, y el programa queda guardado en la tarjeta. A partir de ahora si desconectamos la tarjeta del PC, ésta ejecutará continuamente el programa que hemos instalado en ella.

Si la carga del programa se ha realizado correctamente observaremos que parpadea un led naranja dentro de la tarjeta Arduino. Este led está asociado al pin 13, y nos mostrará la actividad de ese pin siempre que se active/desactive.

PRIMEROS PASOS CON SCRATCH S4A

Para que S4A reconozca la tarjeta Arduino debemos seguir estos pasos:

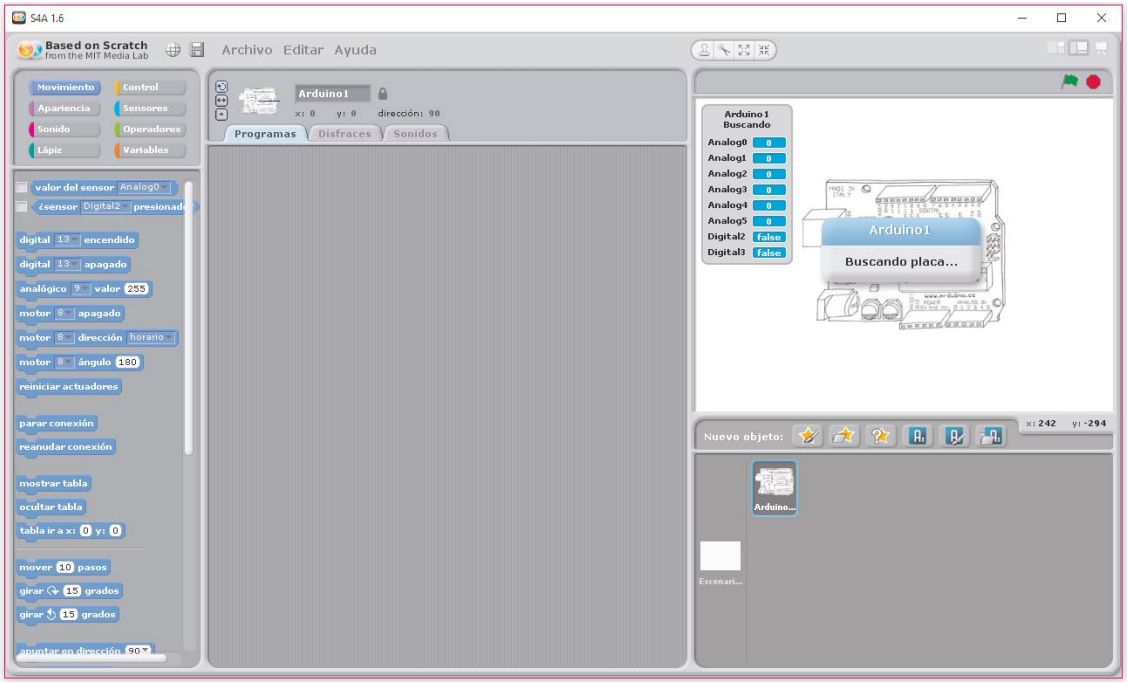
- Abrir el archivo S4AFireware15.ino. Al abrirlo, el sistema Arduino IDE nos indica que debe estar ubicado en una carpeta concreta, aceptamos y automáticamente creará la carpeta y abrirá el archivo.

- Pinchar en la pestaña “Herramientas” y comprobar que la tarjeta seleccionada es Arduino UNO.

- Pinchar en la pestaña “Herramientas” y comprobar que está seleccionado el puerto correcto como indicamos en el apartado anterior.

- Pulsar en cargar y grabar el firmware en la tarjeta Arduino UNO.

- Abrimos S4A y si el proceso se ha hecho correctamente, aparecerá el mensaje “Buscando placa” y en unos segundos, cuando la tarjeta se haya detectado, el mensaje desaparecerá.



Una ventaja del uso de las tarjetas arduino UNO es la posibilidad de programar usando lenguajes técnicos basados en la introducción de líneas de código o usando lenguajes gráficos basados en el uso de bloques, más sencillos y fáciles de comprender para usuarios principiantes y niños.

El uso de sistemas basados en bloques supone eliminar fallos de sintaxis durante la elaboración de los programas, ya que programar con estos sistemas es como realizar puzzles. Las instrucciones están clasificadas en categorías por colores, y tienen diferentes formas. A la hora de realizar un programa, no podremos juntar instrucciones incompatibles, eliminando los errores de sintaxis y permitiendo que el alumno sólo se centre en desarrollar la lógica de la programación. Así, como si hiciésemos un puzzle, juntamos instrucciones que puedan encajar unas con otras atendiendo a formas y colores como si del dibujo del puzzle se tratase.

En este manual nos centramos en 3 lenguajes de programación:

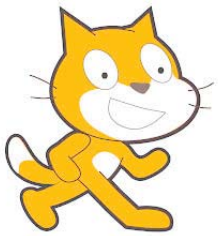
- Scratch S4A
- Arduino IDE (processing)

A lo largo de este capítulo vamos a aprender las bases de estos sistemas de programación.

4.1 SCRATCH S4A

S4A es una modificación del software libre Scratch que nos permite crear programas para Arduino UNO, pero teniendo en cuenta que los proyectos siempre serán dependientes de su conexión con el PC.

Está basado en programación por bloques, y sus instrucciones han sido diseñadas con un lenguaje natural, eliminando términos técnicos y empleando una terminología más natural. Así se facilita el acceso a la programación en niveles educativos básicos.



4. INTRODUCCIÓN A LA PROGRAMACIÓN

CATÁLOGO DE INSTRUCCIONES

Las instrucciones se clasifican por colores y en las siguientes categorías:

- Movimiento (azul oscuro): conjunto de instrucciones relacionadas con el control de los pines de la tarjeta arduino, así como control del movimiento de cualquier personaje que tengamos dentro del escenario de representación.

- Apariencia (morado): instrucciones orientadas a modificar el aspecto de los personajes de nuestra aplicación. Para el caso de arduino, es un conjunto de instrucciones que apenas se utilizan.

- Sonido (rosa): conjunto de instrucciones relacionadas con la elaboración de aplicaciones musicales, emitiendo sonidos y notas musicales.

- Lápiz (verde oscuro): Scratch nos ofrece la posibilidad de que los personajes dejen un rastro en durante sus movimientos por el escenario como si arrastrase un lápiz durante su trayectoria. Este rastro se genera con las instrucciones que podremos encontrar en esta sección.

- Control (Yema): las instrucciones incluidas en esta sección son imprescindibles para crear la lógica de nuestros programas. Incluyen condicionales, bucles y llamadas de acción.

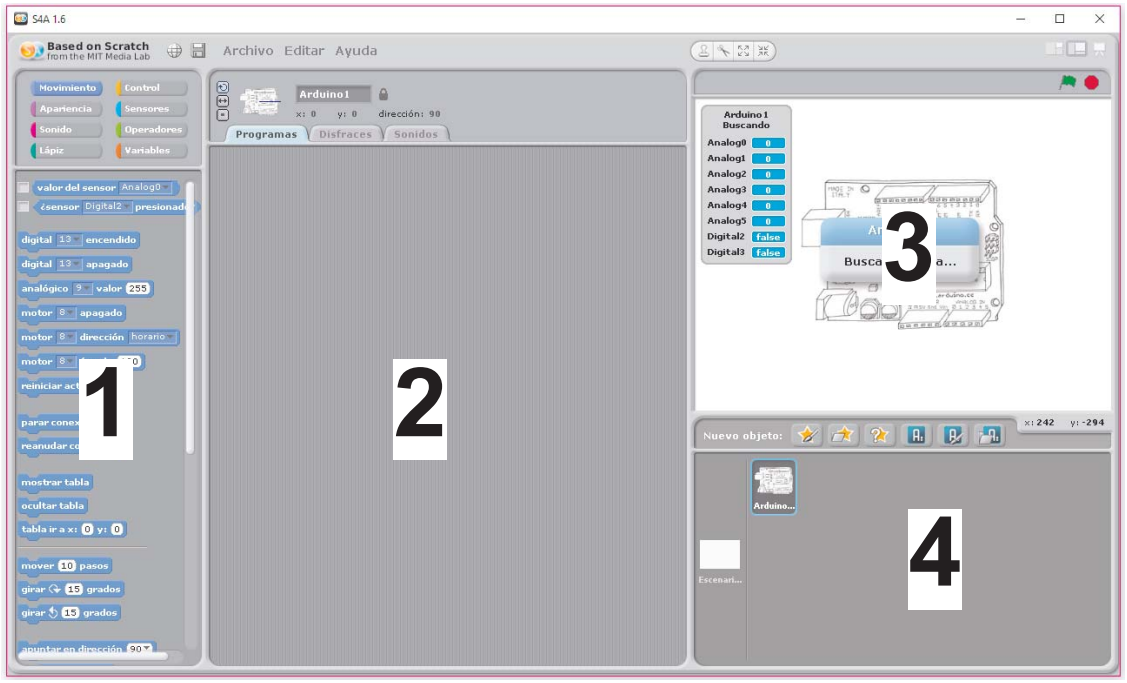
- Sensores (azul): Instrucciones de interacción con el ratón, el teclado, sonidos y los personajes.

- Operadores (verde claro): operaciones matemáticas, lógicas y con cadenas de texto.

- Variables (naranja): instrucciones para el almacenamiento y gestión de datos.

ENTORNO DE PROGRAMACIÓN S4A

El entorno de programación se divide en varias zonas.



Zona 1: conjuntos de instrucciones. Pulsando en cada uno de los botones de la zona superior, en esta zona 1 se despliega todo el catálogo de instrucciones de la categoría seleccionada.

Zona 2: aquí estructuramos nuestro programa arrastrando instrucciones desde la zona 1.

Zona 3: Escenario, donde normalmente veríamos el resultado de nuestro programa cuando éste es virtual. En el caso del trabajo con Arduino UNO LOG 4031, lo normal es olvidarnos de personajes y escenarios por que buscamos ver el resultado directamente en el proyecto que se ha realizado físicamente.

Zona 4: personajes que van a intervenir en nuestro programa. Trabajando con arduino uno, lo más habitual es que nuestro único personaje sea la propia tarjeta controladora.

INSTRUCCIONES DE CONTROL

Al presionar bandera

Todo programa en scratch comienza con un clic de ratón sobre el icono de la bandera, y se paraliza pulsando sobre el icono del círculo rojo.

Esta instrucción será la primera que incluyamos en nuestro proyecto. El resto de instrucciones se irán agregando bajo esta. Si nos fijamos en el dibujo, en la parte inferior de la instrucción encontraremos una pequeña pestaña. Cuando acercamos otra instrucción hacia la pestaña, aparece una línea blanca. Esa línea blanca nos indica que la instrucción se va a añadir al programa en cuanto soltemos el botón del ratón.

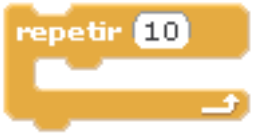
Esperar un segundo

Genera una pausa de un segundo durante la ejecución del programa.

Bucles

Las instrucciones de tipo bucle son aquellas que suponen la repetición de una secuencia de instrucciones. Pueden ser bucles de tipo infinito o bucles que se ejecutan un determinado número de instrucciones.

Genera un bucle infinito dentro del programa. La secuencia de instrucciones que incluyamos dentro de esta instrucción se ejecutará continuamente sin final.



Ejecuta un determinado número de veces las instrucciones contenidas dentro del bucle.

Condicionales

Son instrucciones en las que se evalúa una condición, y en función de si ésta es cierta o falsa, se procede a ejecutar unas instrucciones u otras.

Sólo se ejecutan las instrucciones contenidas dentro del "si" en caso de que sea cierta la condición que se incluye junto al "si". Esta condición normalmente será una comparación de tipo numérico.

En este caso, disponemos de dos juegos de instrucciones. Las instrucciones que introducimos bajo el "si" se ejecutarán en caso de que la condición sea cierta. En caso de que sea falsa, se ejecutarán las instrucciones bajo el "si no".

INSTRUCCIONES DE OPERACIÓN (OPERADORES)

En esta sección encontraremos instrucciones que utilizaremos para realizar operaciones matemáticas y comparaciones de tipo lógico para su uso en condiciones.

Las instrucciones redondeadas son operaciones matemáticas.

Las instrucciones con los bordes puntiagudos son comparaciones que utilizaremos para generar condiciones en bucles e instrucciones condicionales. Pueden enlazarse varias condiciones con operadores de tipo:

- y: deben cumplirse ambas condiciones.
- o: se cumple alguna de las condiciones.
- no: no se da la condición.

INSTRUCCIONES DE MOVIMIENTO

En esta sección se incluyen las instrucciones propias de Arduino UNO.(LOG 4031).

En S4A se ha eliminado el concepto de pines y estados (high / low) y se utiliza un lenguaje más natural para acceder a cada una de las características de la tarjeta controladora.

Encender y apagar un pin digital:

Con esta instrucción encendemos cualquier operador conectado a determinados pines digitales: 13, 12, 11, 10.

Con esta instrucción apagamos cualquier operador conectado a determinados pines digitales: 13, 12, 11, 10.

El resto de pines digitales de tarjeta controladora no podemos utilizarlos con estas dos instrucciones. Han sido reservados para otros usos que veremos más adelante.

Encender y apagar un pin analógico

Con esta instrucción modulamos la señal de un operador conectado a los pines PWM 9, 6 y 5 de la placa Arduino UNO, pudiendo enviar un valor desde 0 hasta 255.

Control de motores

Se ha creado un juego de instrucciones propio para la gestión de cualquier motor conectado a los pines 4, 7 y 8.

Posiciona un servomotor, conectado al pin 4, 7 u 8, en un ángulo determinado.

Paraliza el motor conectado al pin 4, 7 u 8.

Hace girar en un determinado sentido el motor conectado al pin 4, 7 u 8.

VARIABLES

Una variable es un espacio donde guardar un dato. Podríamos definirla como una caja a la que ponemos un nombre y donde guardamos un dato. Cuando queremos acceder al dato, le llamamos por el nombre asignado.

Scratch dispone de un juego de instrucciones que nos permite crear variables, fijar el valor de la variable a un determinado dato, realizar operaciones sobre la variable y mostrar el contenido de la variable.

4.2 PROGRAMACIÓN CON ARDUINO IDE (PROCESSING)

Arduino IDE es un entorno basado en Processing que nos permite la programación de la tarjeta Arduino UNO utilizando el lenguaje processing. Incluye instrucciones y librerías propias que facilitan el proceso de programación.

No obstante, al ser un lenguaje basado en código, sin carácter gráfico ni visual, tenemos que aprender bien la semántica del lenguaje para minimizar los errores por fallos de escritura. Igualmente debemos tener claros cuales son los errores de sintaxis más comunes para solventar los problemas con celeridad.

SINTAXIS DE PROCESSING

Es importante tener claros las siguientes reglas del lenguaje:

- Las instrucciones son sensibles a las mayúsculas y minúsculas. Es decir, si una instrucción está definida dentro de processing con una letra en mayúscula y las demás en minúscula, debemos respetarlo pues en caso contrario no la reconocerá y nos devolverá un mensaje de error.
- Toda línea termina en ; (punto y coma) excepto llaves, bucles y condicionales.
- Las asignaciones numéricas se realizan con el operador = y las comparaciones con == (doble igual).

Para familiarizarnos con la sintaxis vamos a realizar un repaso por las instrucciones más habituales con sencillos ejemplos.

ESTRUCTURA DE UN PROGRAMA PARA ARDUINO

Todo programa para Arduino realizado con processing, consta de la siguiente estructura:

```
void setup() {
        Instrucción;
        Instrucción;
        …

}

void loop() {
        Instrucción;
        Instrucción;
        …
}
```

Setup y loop son instrucciones del sistema que utilizaremos siempre:

- En **setup** configuramos los pines de la tarjeta controladora que vamos a utilizar en nuestro programa, pudiendo ser de entrada o de salida:

```
pinMode(nº_de_pin, INPUT);   Para definir el pin como entrada
pinMode(nº_de_pin, OUTPUT);  Para definir el pin como salida
```

- En **loop** vamos a introducir el programa que queremos ejecutar dentro de la placa. Este programa se ejecutará continuamente como si de un bucle infinito se tratase.

DEFINICIÓN DE VARIABLES

En la realización de nuestros programas puede ser de utilidad la definición y uso de variables.

Una variable es una zona de memoria donde vamos a guardar un dato al cual asignaremos un nombre. Una vez definida cuando queramos hacer referencia al dato lo haremos a través del nombre asignado.

Las variables pueden ser útiles para registra y almacenar por ejemplo los datos proporcionados por un sensor.

Para definir una variable ponemos una línea por encima de void setup el siguiente código:

Int nombre_de_variable = dato;

Donde dato será el número o carácter que emplearemos como dato inicial en la variable.

INSTRUCCIONES PROPIAS DE PROCESSING

A continuación se detallan algunas instrucciones de processing que serán de utilidad en la elaboración de nuestros programas:

Visualización de datos en pantalla

La visualización de datos en pantalla resulta de utilidad para localizar posibles errores en la lógica de nuestro programa. En processing esta tarea se realiza a través del Monitor serial.

Para ello primero hay que configurar el puerto serie, introduciendo la línea:

Serial.begin(9600);

Dentro de la función setup.

Para visualizar un dato en la pantalla utilizamos la instrucción:

Serial.println(dato);

Dentro de la función loop.

Para poder ver los datos que se están enviando a través de println tenemos que abrir el monitor serial: menú herramientas > monitor serial.

Veamos un ejemplo:

```
void setup() {
        Serial.begin(9600);
}

void loop() {
        Serial.println('hola mundo');
}
```

Si abrimos el monitor serial, veremos que nos muestra en pantalla la frase 'hola mundo' continuamente.

Pausas y retardos

delay(Número_de_milisegundos)

Genera una pausa dentro del programa de un determinado tiempo indicado en milisegundos.

Estructura condicional

```
if (condición){
                Instrucciones;
}
else{
                Instrucciones;
}
```

Sentencia condicional, que ejecuta un conjunto de instrucciones u otro en función de si se da la condición indicada. Como ejemplo, podríamos programar una comparación de dos números que nos muestre en pantalla cuál es el mayor:

```
if(A>B){
        Serial.println('A es mayor que B');
}
else{
        Serial.println('B es mayor que A');
}
```

INSTRUCCIONES DE CONTROL DE PINES

La programación de una tarjeta de control Arduino UNO consiste principalmente en interactuar con los operadores conectados a cada uno de sus pines, pudiendo activar o desactivar estos pines así como leer datos que nos proporcionen sensores conectados a la placa.

Para realizar este tipo de acciones disponemos de las siguientes instrucciones:

digitalWrite(nº_de_pin, estado)

digitalWrite(nº_de_pin, HIGH) Activa el operador que tengamos conectado a un determinado pin.

digitalWrite(nº_de_pin, LOW) Desactiva el operador que tengamos conectado a un determinado pin.

Veamos un ejemplo:

```
void setup() {
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH);
  delay(1000);
  digitalWrite(13, LOW);
  delay(1000);
}
```

En la función setup hemos definido el pin número 13 como un pin de salida, al que podremos conectar físicamente operadores de salida como un led o un zumbador.

En la función loop introducimos las instrucciones de nuestro programa:

- activar la salida 13
- esperar 1 segundo
- desactivar la salida 13
- esperar 1 segundo

… y volver a empezar

Así provocamos una intermitencia del dispositivo conectado al pin 13, encendiendo y apagando dicho operador con intervalos de 1 segundo de espera.

analogWrite(nº_de_pin, valor)

Envía una señal de 0 a 255 al operador conectado a alguno de los pines PWM (~) de la placa arduino UNO.

Si enviamos un valor 0, apagaremos el operador y según el valor que enviemos activaremos el operador con mayor o menor señal.

Si conectamos un led al pin número 9 de la placa y realizamos el siguiente programa, veremos que el led se enciende con diferente intensidad de luz en función del valor introducido en la señal analogWrite:

```
void setup() {
  pinMode(9, OUTPUT);
}

void loop() {
  analogWrite(9, 0);
  delay(1000);
  analogWrite(9, 50);
  delay(1000);
  analogWrite(9, 100);
  delay(1000);
  analogWrite(9, 150);
  delay(1000);
  analogWrite(9, 200);
  delay(1000);
  analogWrite(9, 250);
  delay(1000);
}
```

digitalRead(nº_de_pin)

Nos permite leer los datos aportados por un operador conectado a un pin digital. Supongamos que conectamos un pulsador al pin 12. Definiendo ese pin como entrada, podemos leer cuándo ha sido pulsado.

Veamos un ejemplo: encendemos el led del pin 13 siempre que se presione el pulsador del pin 12.

```
void setup() {
  pinMode(12, INPUT);
  pinMode(13, OUTPUT);
}

void loop() {
   if (digitalRead(12)==HIGH){
            digitalWrite(13, HIGH);
   }
   Else{
            digitalWrite(13, LOW);
   }
}
```

Conectamos al pin 12 un pulsador y configuramos el pin como entrada (input) para realizar la lectura de las pulsaciones.

En la función loop indicamos que si la lectura del pulsador nos devuelve el valor HIGH es por que éste fue presionado. En este caso introducimos el valor HIGH en el pin 13, activando su led. En caso contrario, el pin 12 nos ha devuelto el valor LOW (no presionado), así pues procedemos desactivando el led del pin 13 (LOW).

Observar que en la comparación hemos utilizado el operador == (doble igual). Si hubiésemos introducido un solo símbolo = realizaría una asignación, generando siempre una afirmación que no siempre será cierta.

Otra forma de escribir este mismo programa sería la siguiente:

```
void setup() {
  pinMode(12, INPUT);
  pinMode(13, OUTPUT);
}

void loop() {
   digitalWrite(13, digitalRead(12));
}
```

Configuramos el pin 13 con el mismo estado que el pin 12.

Si presionamos el pulsador, nos genera un estado HIGH que introducimos directamente en el pin 13, activando el led.

Si por el contrario nos genera un estado LOW porque el pulsador no fue presionado, guarda este estado en el pin 13, apagando el led.

analogRead(nº_de_pin)

Nos permite leer los datos aportados por un operador conectado a un pin analógico (A0 – A5). Supongamos que conectamos una resistencia de luz LDR (LOG 731)al pin A0, podemos realizar lecturas de ese pin y ver la cantidad de luz detectada. Este pin no requiere de configuración porque los pines desde A0 hasta A5 son todo entradas analógicas.

Veamos un ejemplo: vamos a encender el led del pin 13 siempre que se detecte poca luz (supongamos para un valor de la LDR inferior a 50) a través de una LDR conectada al pin A0.

```
void setup() {
  pinMode(13, OUTPUT);
}

void loop() {
   if (analogRead(0)<50){
            digitalWrite(13, HIGH);
   }
   Else{
            digitalWrite(13, LOW);
   }
}
```

DECLARACIÓN DE LIBRERÍAS

Algunos operadores contienen librerías de instrucciones específicas que facilitan su programación como es el caso de los servomotores, sensor ultrasonido, display LCD….

Para incluir esas librerías introducimos una línea al principio del archivo de nuestro programa (seria la primera línea del archivo) como la siguiente:

#include <libreria.h>

Donde la palabra librería será sustituido por el nombre de la librería en cuestión a utilizar.

Una vez incluida esa línea podremos utilizar dentro de nuestro programa cualquier instrucción que se incluya en la librería.

A lo largo de las prácticas incluidas en el manual de prácticas conoceremos diferentes librerías que podemos utilizar con los operadores mencionados.

5. PRÁCTICAS

5.1. LUZ INTERMITENTE

Introducción

Sencillo proyecto consistente en un LED que parpadea continuamente.

Para ello debemos conectar un LED al pin 13 de la tarjeta controladora.

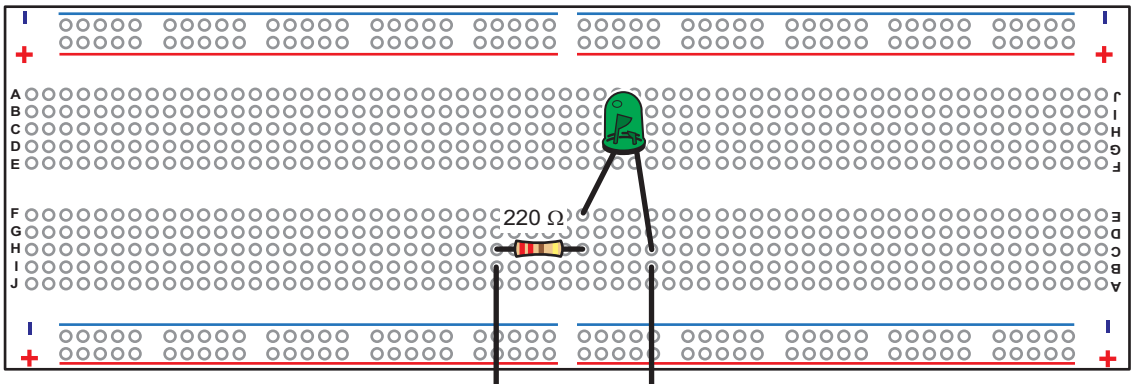
El LED es un operador con una determinada polaridad que debemos respetar. Así pues, conectaremos el LED siguiendo estas indicaciones:

- La pata del LED larga (+) al pin 13 de la placa.
- La pata del LED corta (-) se conecta a la resistencia.
- La resistencia se conecta al pin GND de la placa.

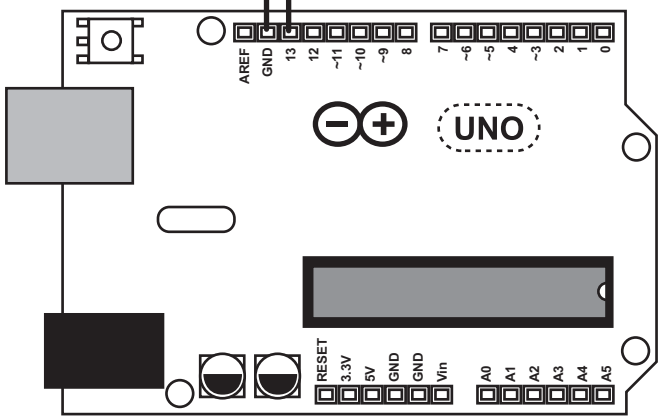
Lista de materiales con placa board

- 1 Tarjeta arduino LOG 4031
- 1 Cable USB LOG 4009
- 1 Protoboard LOG 887
- 1 Led verde LOG 722
- 1 Resistencia de 220 ohmios (rojo-rojo-marrón) LOG 748 220
- 2 Latiguillos board macho-macho

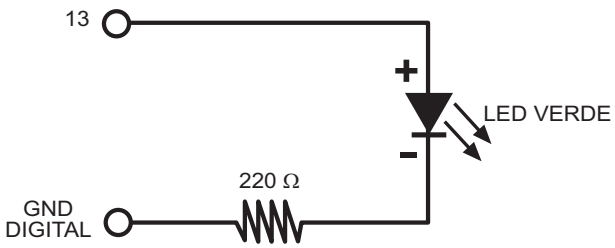
Montaje en placa board



- Pin GND a la resistencia
- Pin 13 al diodo LED



Esquema Eléctrico



Esquema del programa

Inicio de programa
    Bucle infinito
        activa salida 13 (enciende el LED)
        pausa
        desactiva salida 13 (apaga el LED)
        pausa
    Fin del bucle
fin de programa

Programa con Scratch

Programa con processing

```
int LED = 13;

void setup() {
    pinMode(LED, OUTPUT); //Se configura el pin 13 como salida
}

void loop() {
    digitalWrite(LED, HIGH);      // Se enciende el LED
    delay(1000);                  // Se espera unos segundos
    digitalWrite(LED, LOW);       // Se apaga el LED
    delay(1000);                  // Se espera unos segundos
}
```

5.2. AVISO PARA INVIDENTES

Introducción

El circuito consiste en conectar un zumbador al pin 13 de la tarjeta controladora.

El zumbador es un operador con una determinada polaridad que debemos respetar. Así pues, conectaremos el zumbador siguiendo estas indicaciones:

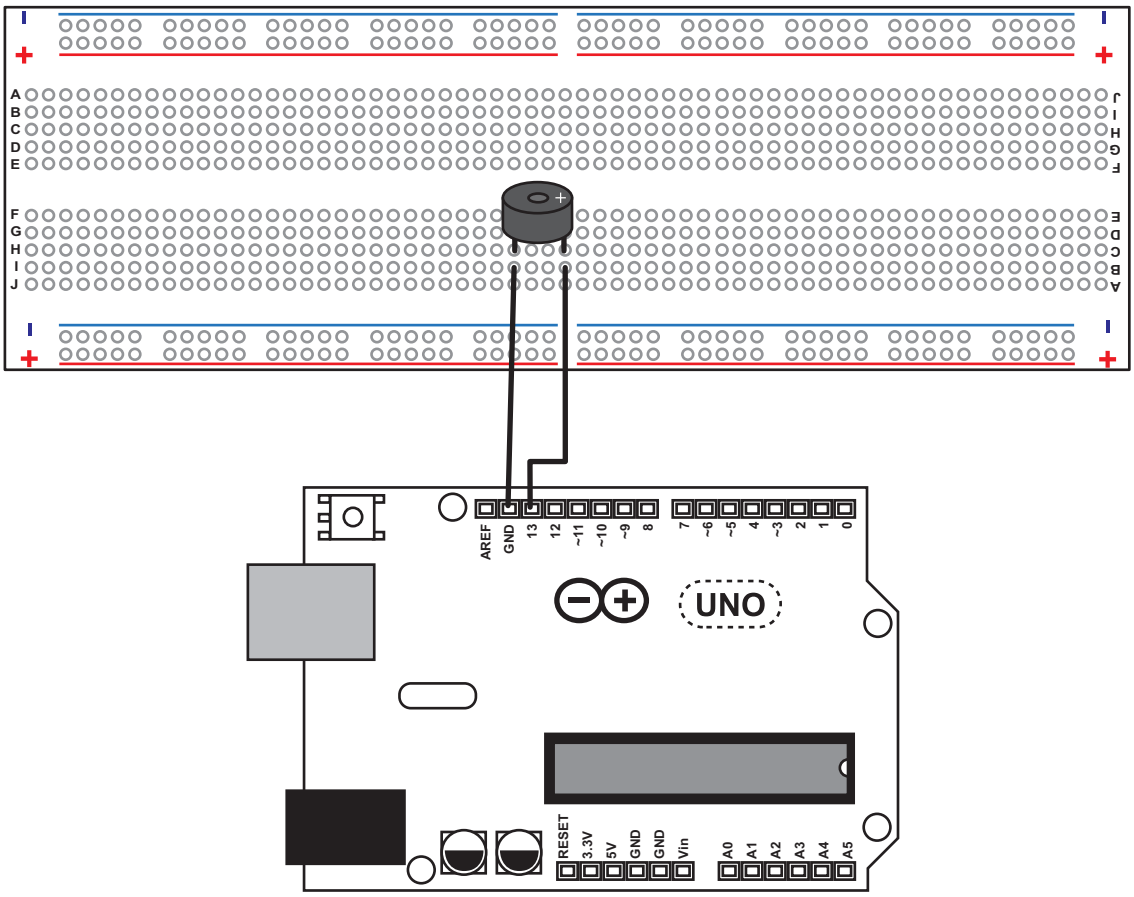
- El cable rojo del zumbador (+) al pin 13 de la placa.
- El cable negro del zumbador (-) al pin GND de la placa.

Lista de materiales con placa board

- 1 Tarjeta arduino LOG 4031
- 1 Cable USB LOG 4009
- 1 Protoboard LOG 887
- 1 Zumbador piezo-eléctrico LOG 7714
- 2 Latiguillos board macho-macho

Montaje en placa board

- Pin GND al cable negro.
- Pin 13 al cable rojo.



Esquema Eléctrico

Esquema del programa

Inicio de programa
    Bucle infinito
        activa salida 13 (enciende el Zumbador)
        pausa
        desactiva salida 13 (apaga el Zumbador)
        pausa
    Fin del bucle
fin de programa

Programa con Scratch

Programa con processing

```
int Zumbador = 13;

void setup() {
    pinMode(Zumbador, OUTPUT);      //Se configura el pin 13 como salida
}

void loop() {
    digitalWrite(Zumbador, HIGH);   // Se enciende el Zumbador
    delay(1000);                    // Se espera unos segundos
    digitalWrite(Zumbador, LOW);    // Se apaga el Zumbador
    delay(1000);                    // Se espera unos segundos
}
```

5.3. SEMÁFORO

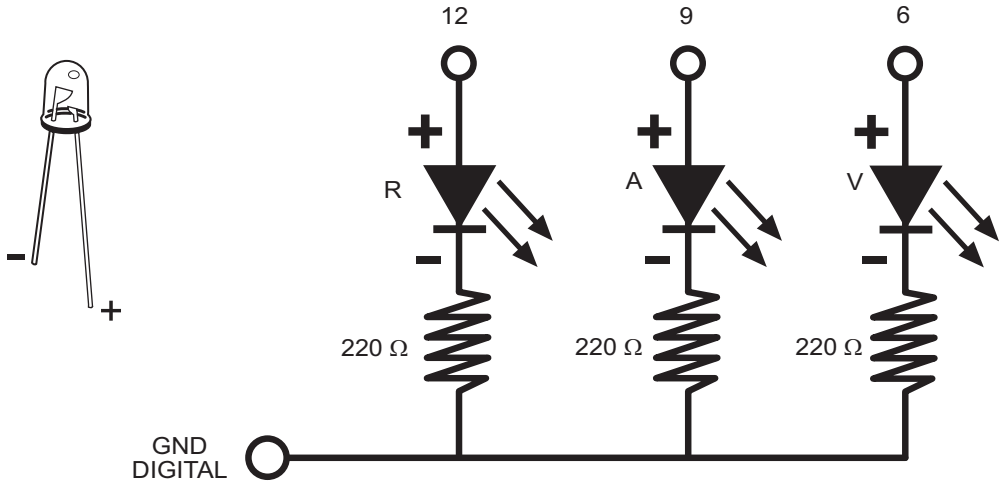
Introducción

Construcción de un semáforo de 3 colores para vehículos con tres diodos LED (verde, amarillo y rojo).

Lista de materiales con placa board

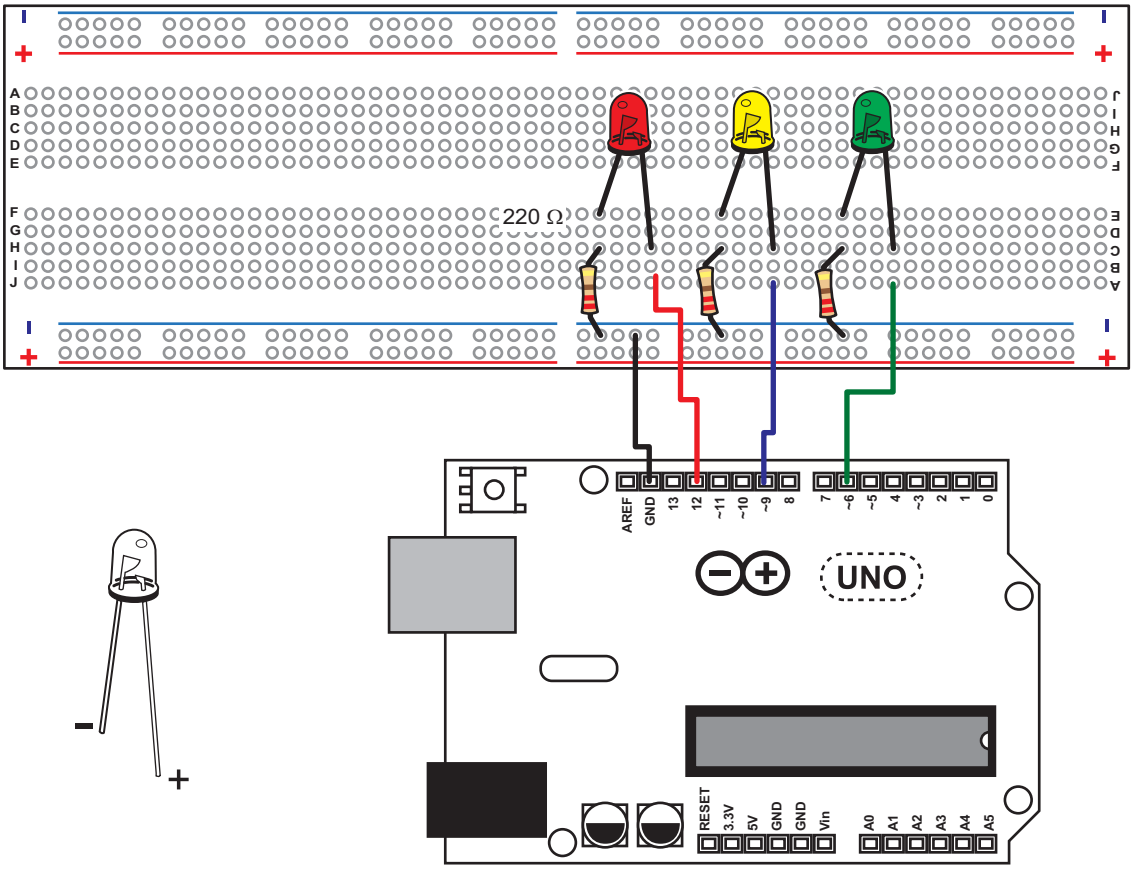
- 1 Tarjeta arduino LOG 4031
- 1 Cable USB LOG 4009
- 1 Protoboard LOG 887
- 1 Led rojo LOG 724
- 1 Led amarillo LOG 723
- 1 Led verde LOG 722
- 3 Resistencias de 220 ohmios (rojo-rojo-marrón) LOG 748 220
- 4 Latiguillos board macho-macho LOG 9519

Esquema Eléctrico



Montaje en placa board

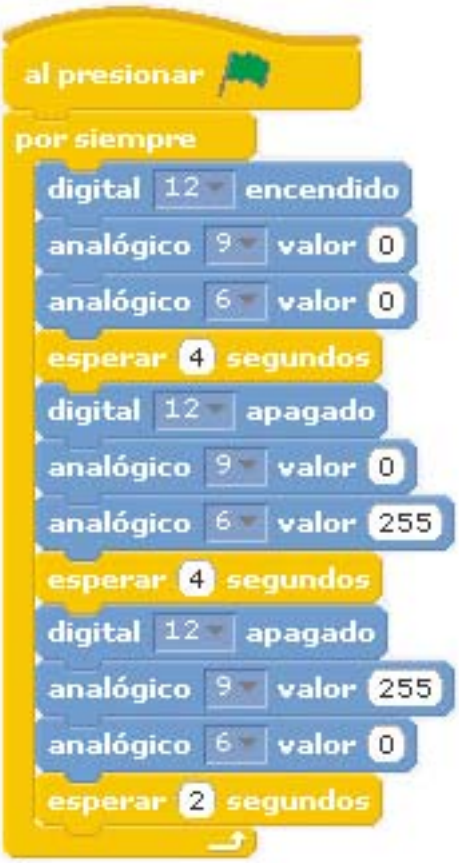
- Pin 6 al diodo LED verde
- Pin 9 al diodo LED amarillo.
- Pin 12 al diodo LED rojo.
- Pin GND a la regleta de conexiones.
- De la regleta de conexiones a las resistencias.



Esquema del programa

Inicio de programa
    bucle infinito
        Activamos la salida 12 (enciende el LED rojo)
        Desactivamos la salida 9 (apaga el LED amarillo)
        Desactivamos la salida 6 (apaga el LED verde)
        pausa
        Desactivamos la salida 12 (apaga el LED rojo)
        Desactivamos la salida 9 (apaga el LED amarillo)
        Activamos la salida 6 (enciende el LED amarillo)
        pausa
        Desactivamos la salida 12 (apaga el LED rojo)
        Activamos la salida 9 (enciende el LED verde)
        Desactivamos la salida 6 (apaga el LED verde)
        pausa
    fin del bucle
fin

Programa con Scratch



Programa con processing

```
int LEDR = 12;
int LEDA = 9;
int LEDV = 6;

void setup() {
        pinMode(LEDR, OUTPUT);
        pinMode(LEDA, OUTPUT);
        pinMode(LEDV, OUTPUT);
}
void loop() {
        digitalWrite(LEDR, HIGH);          // Semáforo en rojo
        digitalWrite(LEDA, LOW);
        digitalWrite(LEDV, LOW);
        delay(4000);
        digitalWrite(LEDR, LOW);           // semáforo en verde
        digitalWrite(LEDA, LOW);
        digitalWrite(LEDV, HIGH);
        delay(4000);
        digitalWrite(LEDR, LOW);
        digitalWrite(LEDA, HIGH);
        digitalWrite(LEDV, LOW);           // semáforo en amarillo
        delay(800);
}
```

5.4. SEMÁFORO CON AVISO PARA INVIDENTES

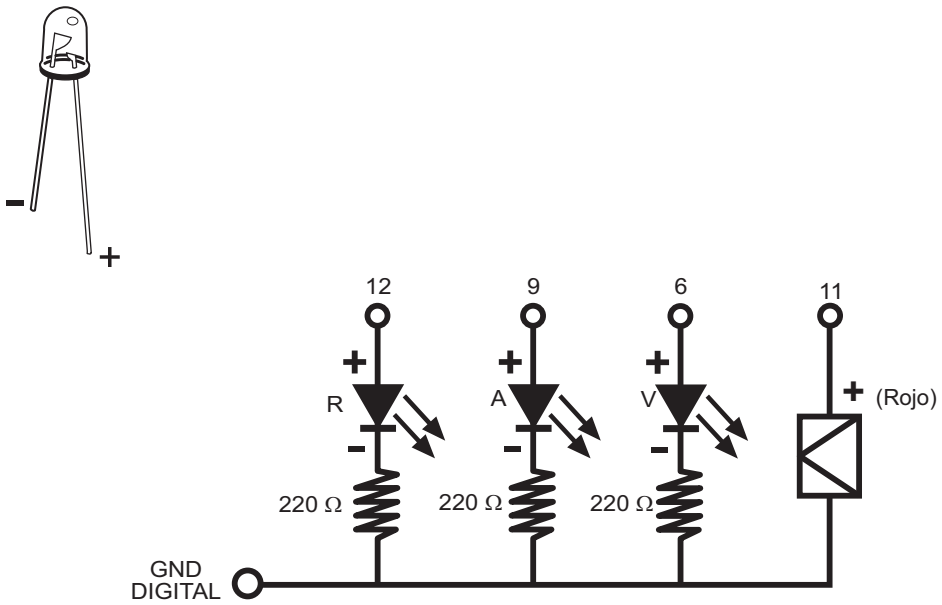
Introducción

Construcción de un semáforo de 3 colores para vehículos con una señal acústica que avisa cuando el semáforo está rojo para que los peatones invidentes puedan cruzar la calzada.

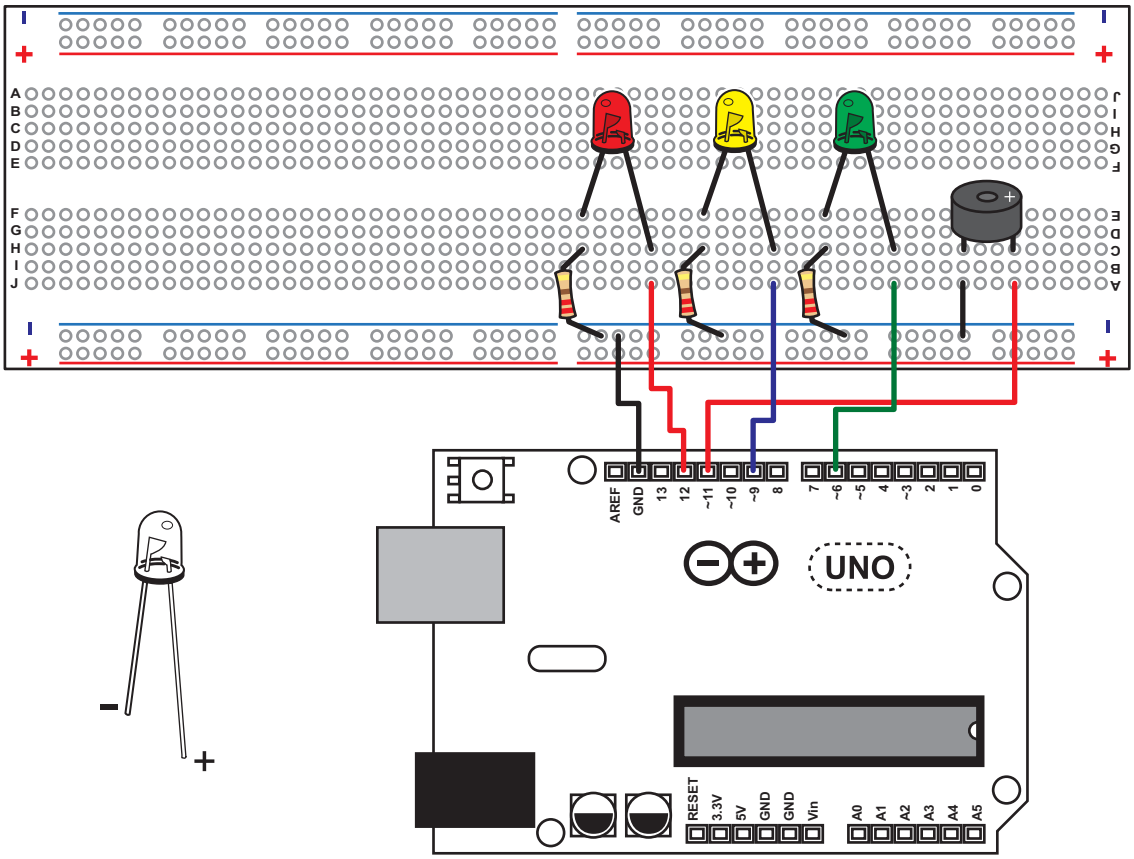
Lista de materiales con placa board

- 1 Tarjeta arduino LOG 4031
- 1 Cable USB LOG 4009
- 1 Protoboard LOG 887
- 1 Led rojo LOG 724
- 1 Led amarillo LOG 723
- 1 Led verde LOG 722
- 3 Resistencias de 220 ohmios (rojo-rojo-marrón) LOG 748 220
- 1 Zumbador piezoeléctrico LOG 7714
- 8 Latiguillos board macho-macho LOG 9519

Esquema Eléctrico



Montaje con placa board



Esquema del programa

Inicio de programa
    bucle infinito
        Activa la salida 12 (enciende el LED rojo)
        Activa la salida 11 (enciende el zumbador)
        Desactiva la salida 9 (apaga el LED amarillo)
        Desactiva la salida 6 (apaga el LED verde)
        Espera 1 minuto
        Desactiva la salida 12 (apaga el LED rojo)
        Desactiva la salida 11 (apaga el zumbador)
        Desactiva la salida 9 (apaga el LED amarillo)
        Activa la salida 6 (enciende el LED verde)
        Espera 1 minuto
        Desactiva la salida 12 (apaga el LED rojo)
        Desactiva la salida 11 (apaga el zumbador)
        Activa la salida 9 (enciende el LED amarillo)
        Desactiva la salida 6 (apaga el LED verde)
        Espera 10 segundos
    fin del bucle
fin

Programa con Scratch

Programa con processing

```
int LEDR = 12;
int LEDA = 9;
int LEDV = 6;
int zumbador = 11;

void setup() {
        pinMode(LEDR, OUTPUT);
        pinMode(LEDA, OUTPUT);
        pinMode(LEDV, OUTPUT);
        pinMode(zumbador, OUTPUT);
}

void loop() {
        digitalWrite(LEDR, HIGH);              // Semafoto en rojo
        digitalWrite(zumbador, HIGH);          // Aviso a invidentes
        digitalWrite(LEDA, LOW);
        digitalWrite(LEDV, LOW);
        delay(4000);
        digitalWrite(LEDR, LOW);               // Semáforo  verde
        digitalWrite(zumbador, LOW);           // Aviso a invidentes
        digitalWrite(LEDA, LOW);
        digitalWrite(LEDV, HIGH);
        delay(4000);
        digitalWrite(LEDR, LOW);               // Semáforo  ámbar
        digitalWrite(zumbador, LOW);
        digitalWrite(LEDA, HIGH);
        digitalWrite(LEDV, LOW);
        delay(800);
}
```

5.5 SEMÁFORO CON PULSADOR

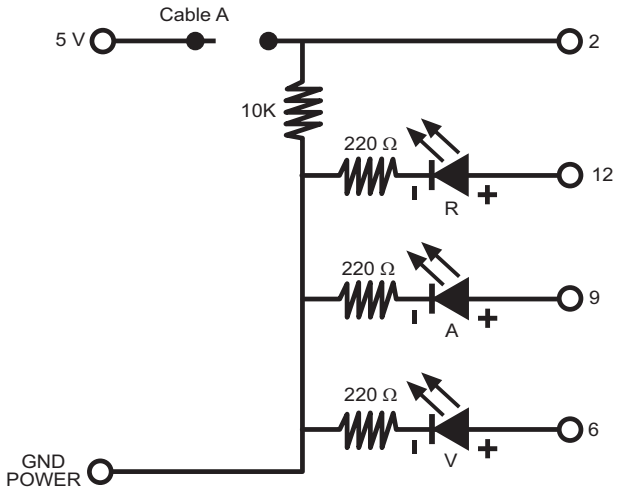
Introducción

Construcción de un semáforo de 3 colores para vehículos que siempre está en verde a no ser que un peatón pulse un botón (conectando el cable A), en cuyo caso realiza el ciclo ámbar-rojo para que el peatón pueda cruzar la calzada.

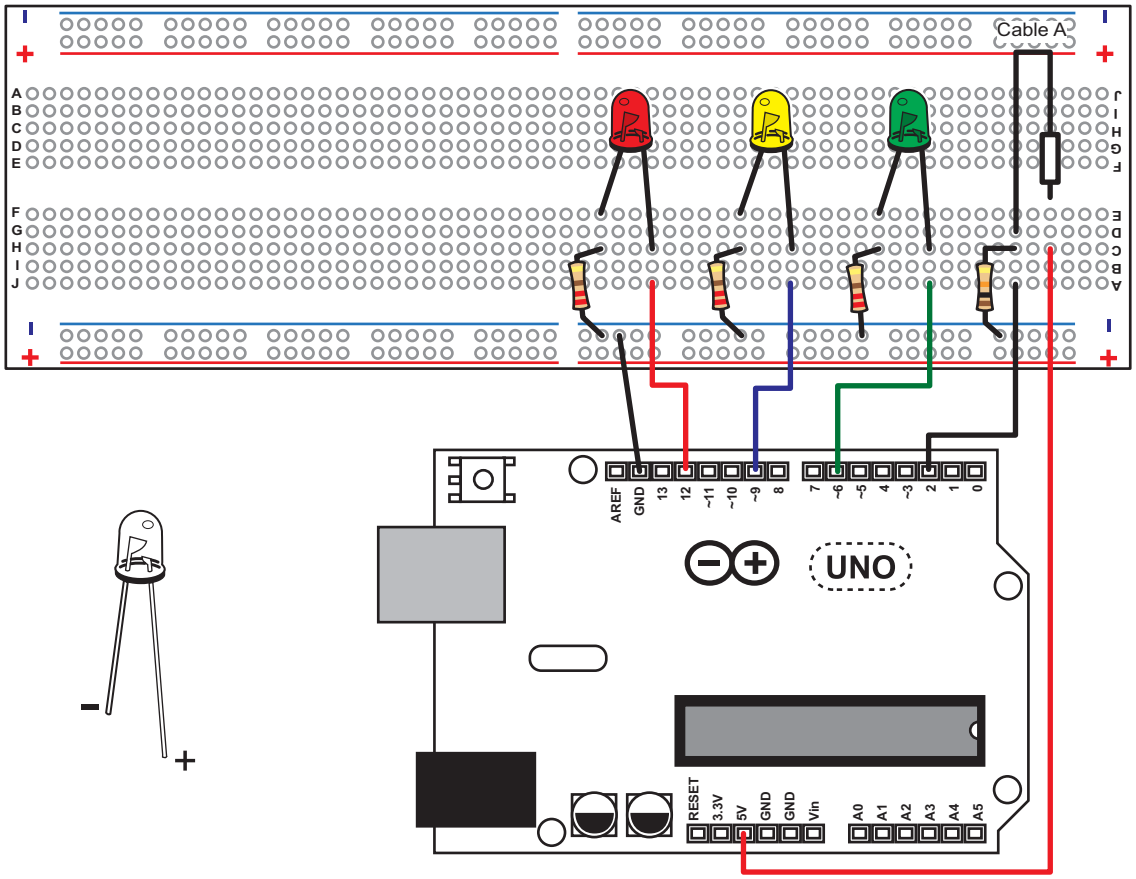
Lista de materiales con placa board

- 1 Tarjeta arduino LOG 4031
- 1 Cable USB LOG 4009
- 1 Protoboard LOG 887
- 1 Led rojo LOG 724
- 1 Led amarillo LOG 723
- 1 Led verde LOG 722
- 3 Resistencias de 220 ohmios (rojo-rojo-marrón) LOG 748 220
- 5 Latiguillos board macho-macho LOG 9519

Esquema Eléctrico



Montaje con placa board



Esquema del programa

```
Inicio de programa
    bucle infinito
        Activa la salida 6              (enciende el LED verde)
            Si esta activa la salida 2  (si pulsamos el pulsador)
                desactiva la salida 6   (apaga el LED verde)
                activa la salida 9      (enciende el LED amarillo)
                espera 10 segundos
                desactiva la salida 9   (apaga el LED amarillo)
                activa la salida 12     (enciende el LED rojo)
                espera 1 minuto
                desactiva la salida 12  (apaga el LED rojo)
            Fin del si
    fin del bucle
fin
```

Programa con Scratch

Programa con processing

```
int LEDR=12;
int LEDA=9;
int LEDV=6;
int pulsador=2;

void setup()
{
        pinMode(LEDR,OUTPUT);
        pinMode(LEDV,OUTPUT);
        pinMode(LEDA,OUTPUT);
        pinMode(pulsador,INPUT);
}

void loop()
{
        digitalWrite(LEDV,HIGH);            //LED verde encendido
        if (digitalRead(pulsador) == HIGH) //lee pulsador
        {
                digitalWrite(LEDV,LOW);    //LED verde apagado
                digitalWrite(LEDA,HIGH);   //LED amarillo encendido
                delay(400);
                digitalWrite(LEDA,LOW);    //LED amarillo apagado
                digitalWrite(LEDR,HIGH);   //LED rojo encendido
                delay(4000);
                digitalWrite(LEDR,LOW);    //LED rojo apagado
        }
}
```

5.6. CRUCE DE SEMÁFOROS

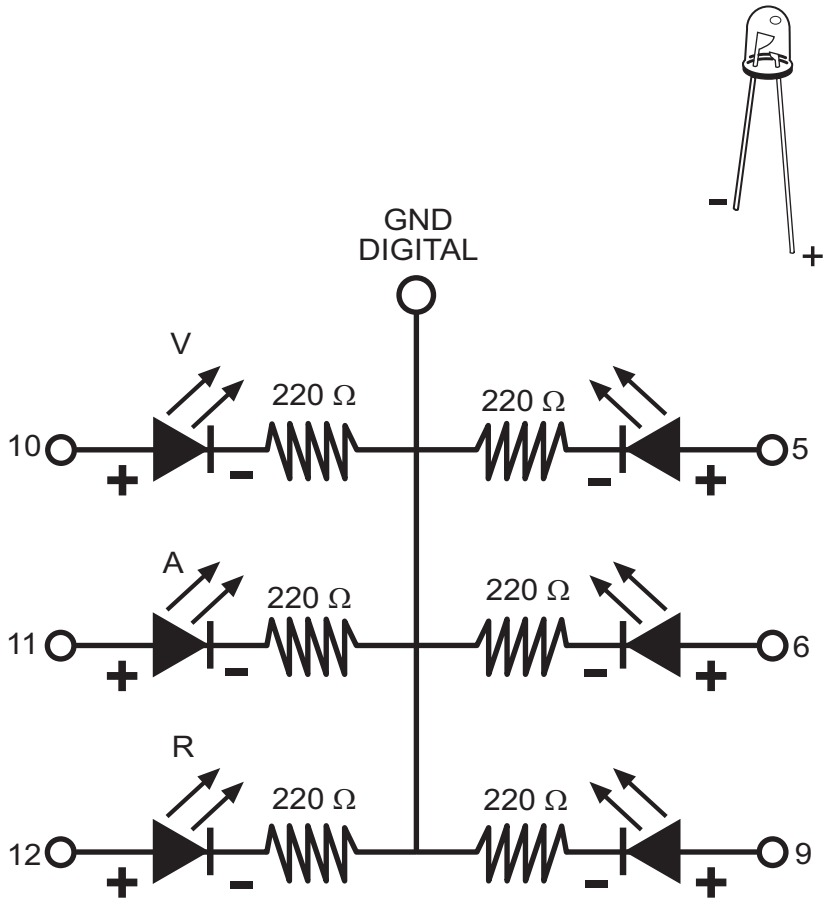
Introducción

Simulación de un cruce de semáforos de 3 colores para vehículos.

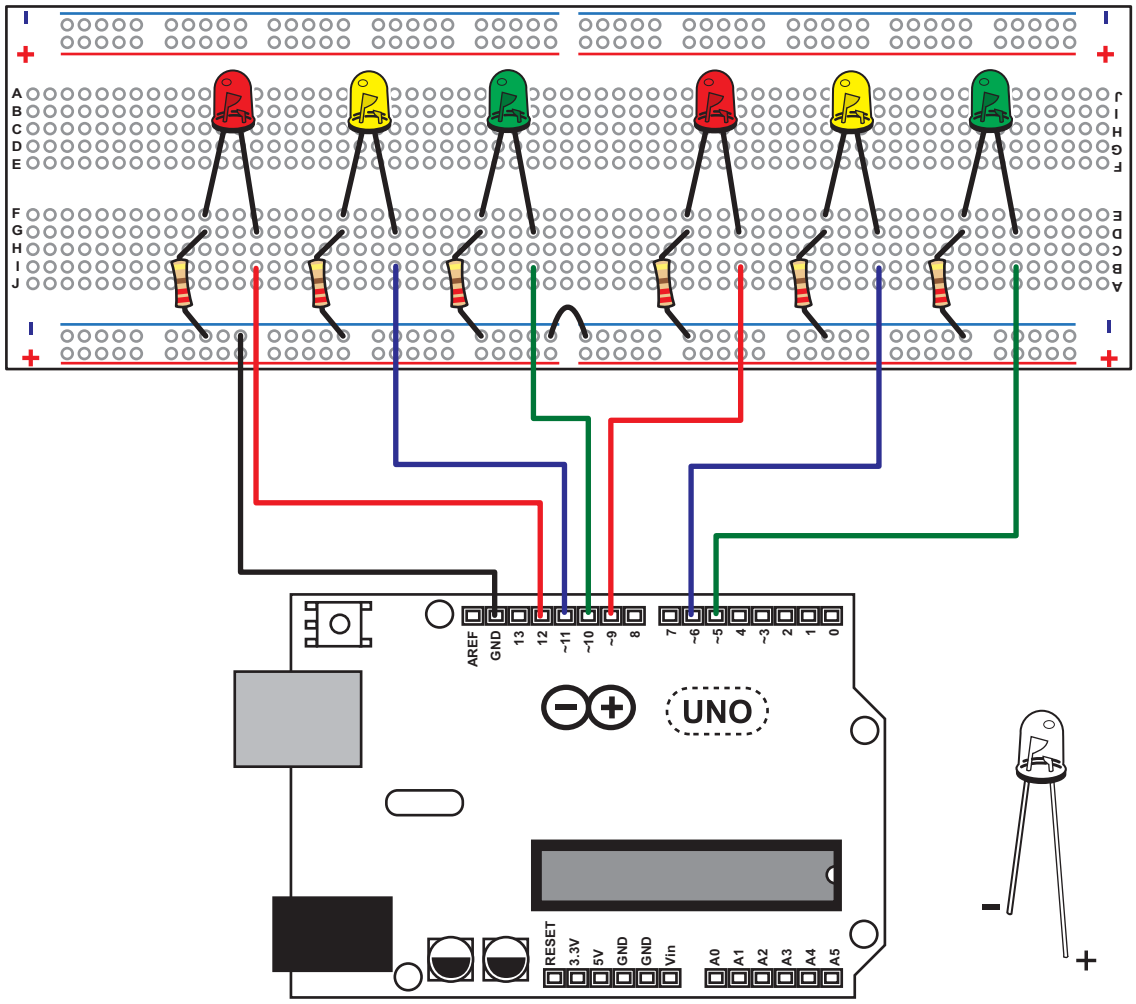
Lista de materiales con placa board

- 1 Tarjeta arduino LOG 4031
- 1 Cable USB LOG 4009
- 1 Protoboard LOG 887
- 6 Resistencias de 220 ohmios (rojo-rojo-marrón) LOG 748 220
- 2 Led rojo LOG 724
- 2 Led amarillo LOG 723
- 2 Led verde LOG 722
- 13 Latiguillos LOG 9519

Esquema del circuito



Montaje en placa board



Programa con Scratch

Programa con Processing

```
int ledR = 12;
int ledA = 11;
int ledV = 10;
int ledR_ = 9;
int ledA_ = 6;
int ledV_ = 5;

void setup() {
        pinMode(ledR, OUTPUT);
        pinMode(ledA, OUTPUT);
        pinMode(ledV, OUTPUT);
        pinMode(ledR_, OUTPUT);
        pinMode(ledA_, OUTPUT);
        pinMode(ledV_, OUTPUT);

}
void loop() {
        digitalWrite(ledR, HIGH);   // rojo verde
        digitalWrite(ledA, LOW);
         digitalWrite(ledV, LOW);
         digitalWrite(ledR_, LOW);
         digitalWrite(ledA_, LOW);
         digitalWrite(ledV_, HIGH);
         delay(4000);
         digitalWrite(ledR, HIGH);   // rojo ambar
         digitalWrite(ledA, LOW);
         digitalWrite(ledV, LOW);
        digitalWrite(ledR_, LOW);
        digitalWrite(ledA_, HIGH);
        digitalWrite(ledV_, LOW);
        delay(800);
        digitalWrite(ledR, LOW);   // verde rojo
        digitalWrite(ledA, LOW);
        digitalWrite(ledV, HIGH);
        digitalWrite(ledR_, HIGH);
        digitalWrite(ledA_, LOW);
        digitalWrite(ledV_, LOW);
        delay(4000);
        digitalWrite(ledR, LOW);   // ambar rojo
        digitalWrite(ledA, HIGH);
        digitalWrite(ledV, LOW);
        digitalWrite(ledR_, HIGH);
        digitalWrite(ledA_, LOW);
        digitalWrite(ledV_, LOW);
        delay(800);
}
```

5.7. ENCENDIDO NOCTURNO

Introducción

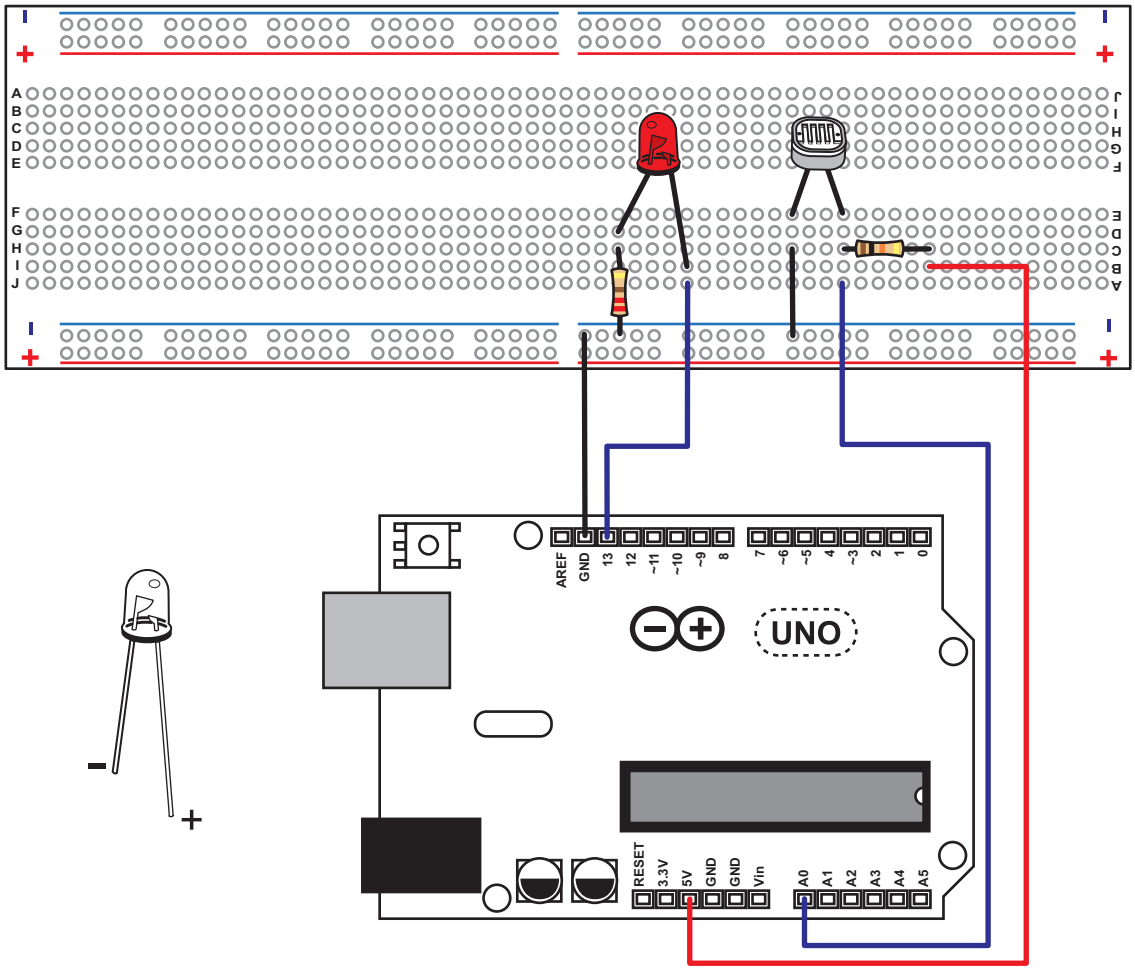
Construcción consistente en una luz que se enciende ante la falta de luz natural.

Una resistencia LDR se encarga de medir la luz ambiente.

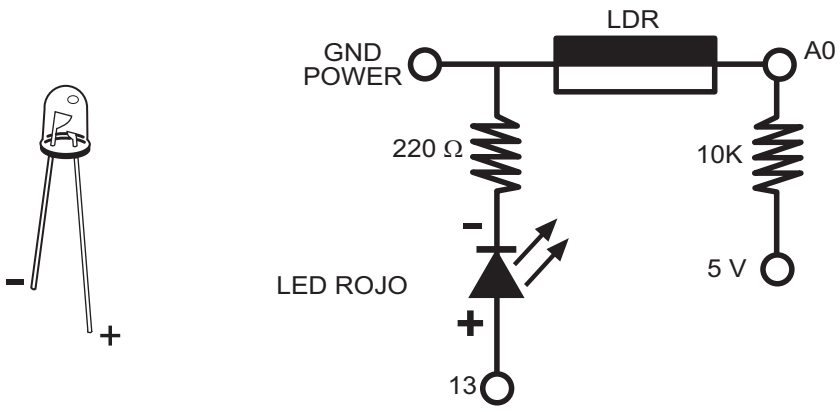
Lista de materiales con placa board

- 1 Tarjeta arduino LOG 4031
- 1 Cable USB LOG 4009
- 1 Protoboard LOG 887
- 1 Led rojo LOG 724
- 1 Resistencia de 220 ohmios (rojo-rojo-marrón) LOG 748 220
- 1 Resistencia de 10K Ohmios (marrón-negro-naranja) LOG 748 10K
- 1 LdR LOG 731
- 5 Latiguillos board macho-macho LOG 9519

Esquema del circuito



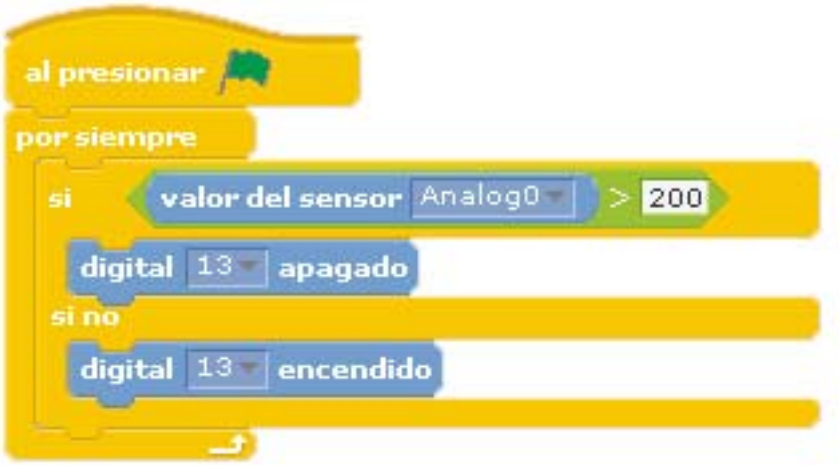
Esquema del circuito



Esquema del programa

```
Inicio de programa
    bucle infinito
        Si valor de entrada analogica A0 < 200     (si el valor de la LDR < 200)
                Activa la salida 13                        (enciende el LED rojo)
        Si no
                Desactiva la salida 13             (apaga el LED rojo)
    fin del bucle
fin
```

Programa con Scratch



Programa con processing

```
int sensorPin = A0;
int LEDPin = 13;
int sensorValue = 0;

void setup() {
        pinMode(LEDPin, OUTPUT);
}

void loop() {
        sensorValue = analogRead(sensorPin);
        if (sensorValue >200){                  // Si LDR < 200
                digitalWrite(LEDPin, LOW);}     // Enciende el LED
        else{
                digitalWrite(LEDPin, HIGH);}    // Apaga el LED

}
```

5.8. CADENA DE MONTAJE

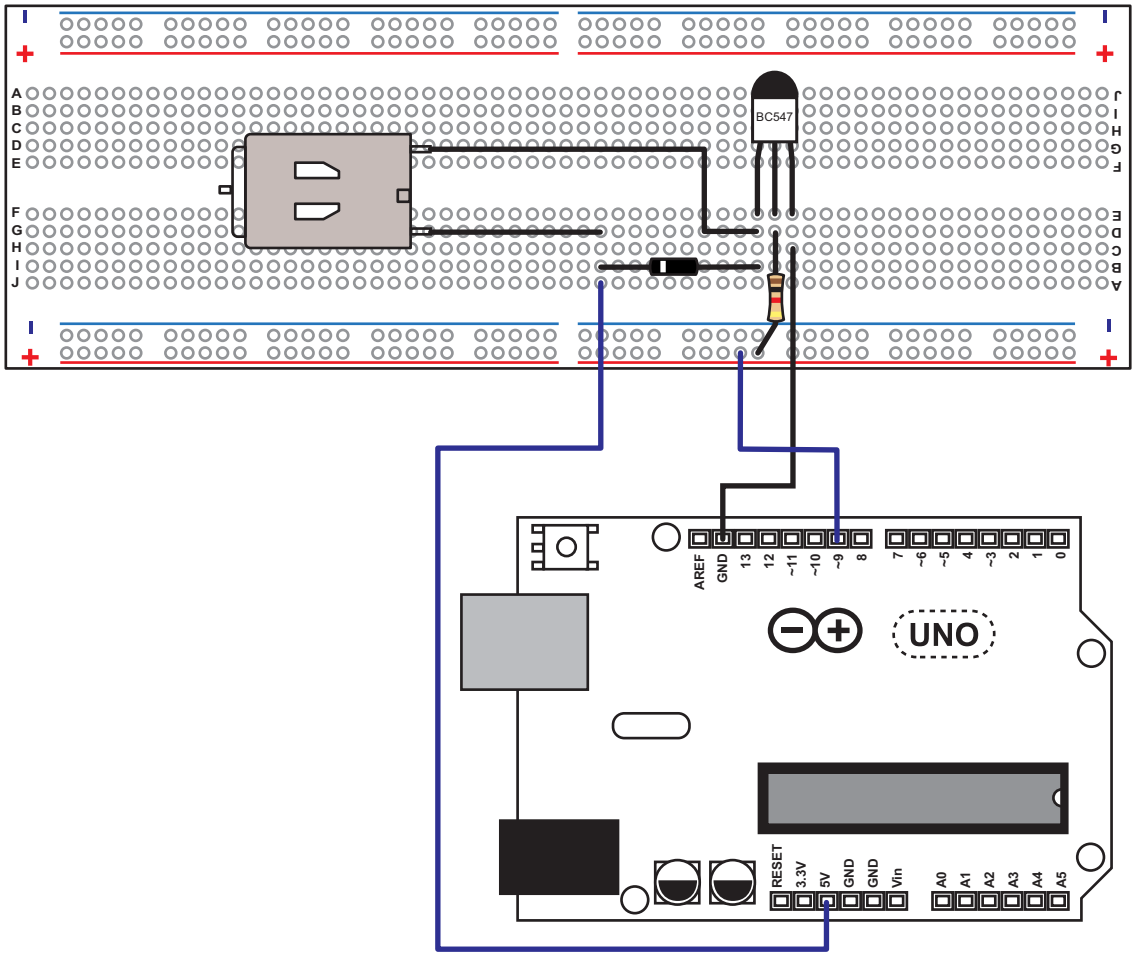
Introducción

Construcción consistente en un motor que realiza el movimiento de avance y pausa alterno de una cadena de montaje.

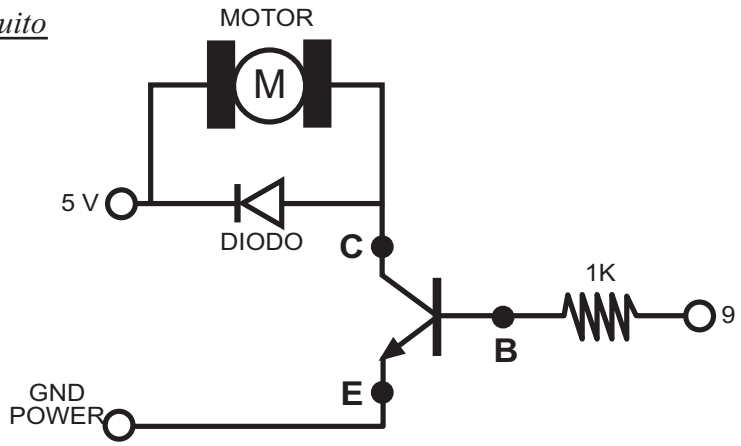
Lista de materiales con placa board

- 1 Tarjeta arduino LOG 4031
- 1 Cable USB LOG 4009
- 1 Protoboard LOG 887
- 1 Motor LOG 02
- 1 Resistencia de 1K Ohmios (marrón-negro-rojo) LOG 748 1K
- 1 Diodo LOG 760
- 1 Transistor NPN BC547 LOG 751
- 8 Latiguillos board macho-macho LOG 9519

Montaje en placa board



Esquema del circuito



Esquema del programa

```
Inicio de programa
        bucle infinito
            Desactiva la salida 9    // Para el motor
            Espera
            Activa la salida 9       // Acciona el motor
            Espera
        fin del bucle
fin
```

Programa con Scratch

Programa con processing

```
int LEDPin = 9;

void setup()
{
        pinMode(LEDPin, OUTPUT);
}

void loop()
{
        digitalWrite(LEDPin, LOW);       // Para el motor
        delay(100);
        digitalWrite(LEDPin, HIGH);      // Activa el motor
        delay(100);
}
```

5.9. VENTILADOR AUTOMÁTICO

Introducción

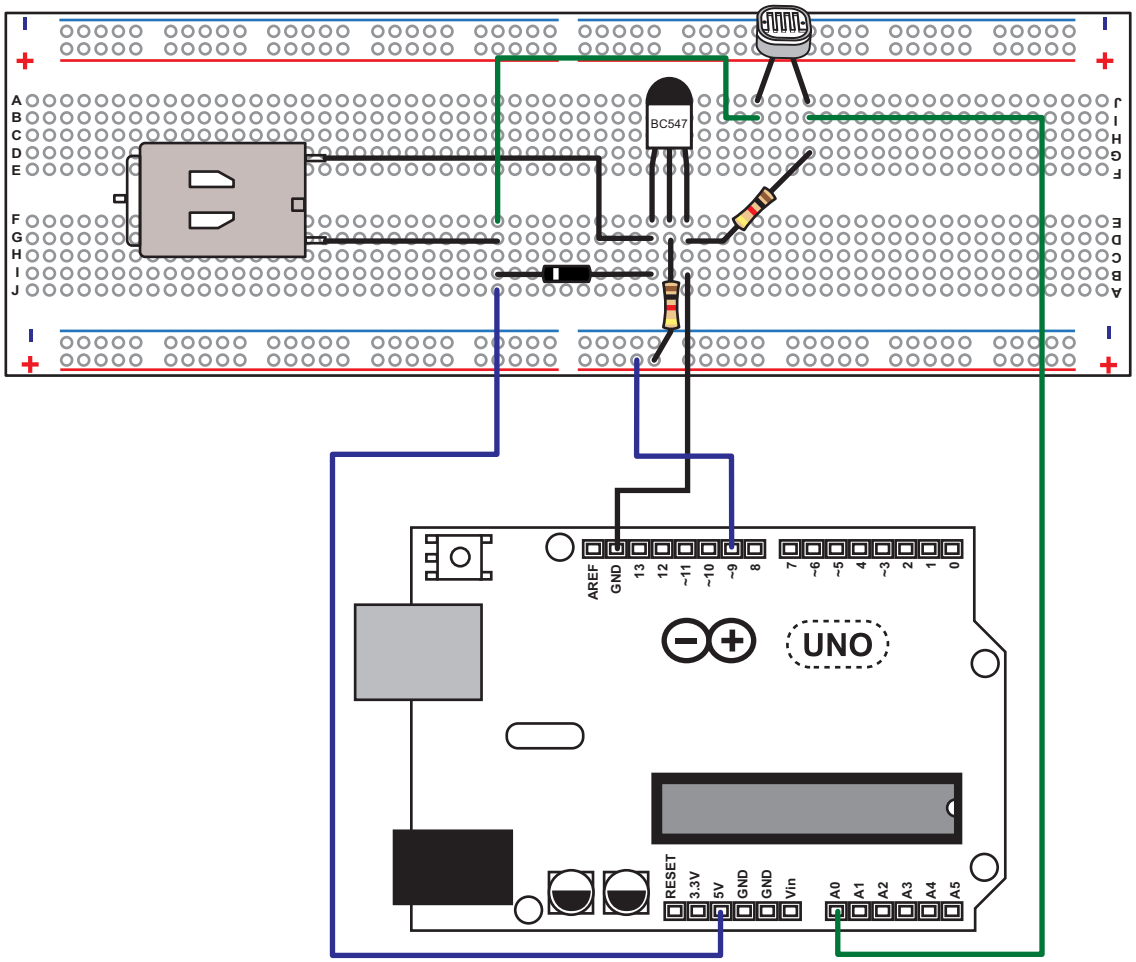
Construcción consistente en un motor que se activa ante la falta de luz.

Utilizamos una LDR para realizar la medición de luz.

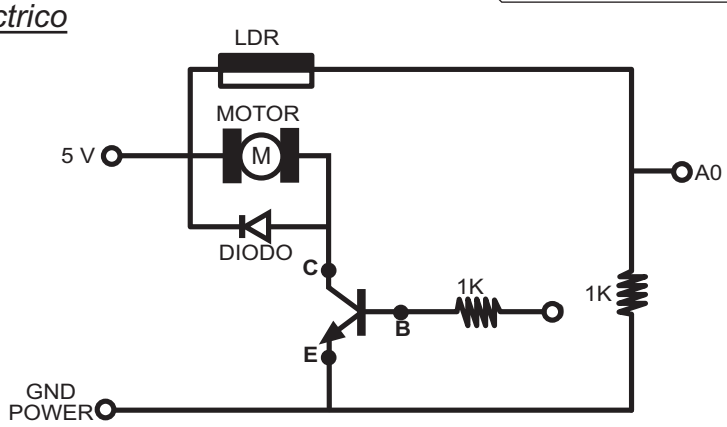
Lista de materiales con placa board

- 1 Tarjeta arduino LOG 4031
- 1 Cable USB LOG 4009
- 1 Protoboard LOG 887
- 1 LDR LOG 731
- 2 Resistencia de 1K (marrón-negro-rojo) LOG 748 1K
- 1 Diodo LOG 760
- 1 Transistor NPN BC547 LOG 751
- 1 Motor LOG 02
- 5 Latiguillos board macho-macho LOG 9519

Montaje en placa board



Esquema eléctrico



Esquema del programa

Inicio de programa
    bucle infinito
        Si valor de entrada analogica > 200 (Si el valor de LDR >200)
            Desactiva la salida 9   (Para motor)
        Si no
            Activa la salida 9   (Activa motor)
    fin del bucle
fin

Programa con Scratch

Programa con processing

```
int sensorPin = A0;
int LEDPin = 9;
int sensorValue = 0;

void setup() {
      pinMode(LEDPin, OUTPUT);
}

void loop() {

        sensorValue = analogRead(sensorPin);          // Si LDR > 200
        if (sensorValue >200){
                digitalWrite(LEDPin, LOW);}           // Apaga el motor
        else{
                digitalWrite(LEDPin, HIGH);}          // Activa motor
}
```

5.10. LUZ ROJA, VERDE Y AZUL (RGB)

Introducción

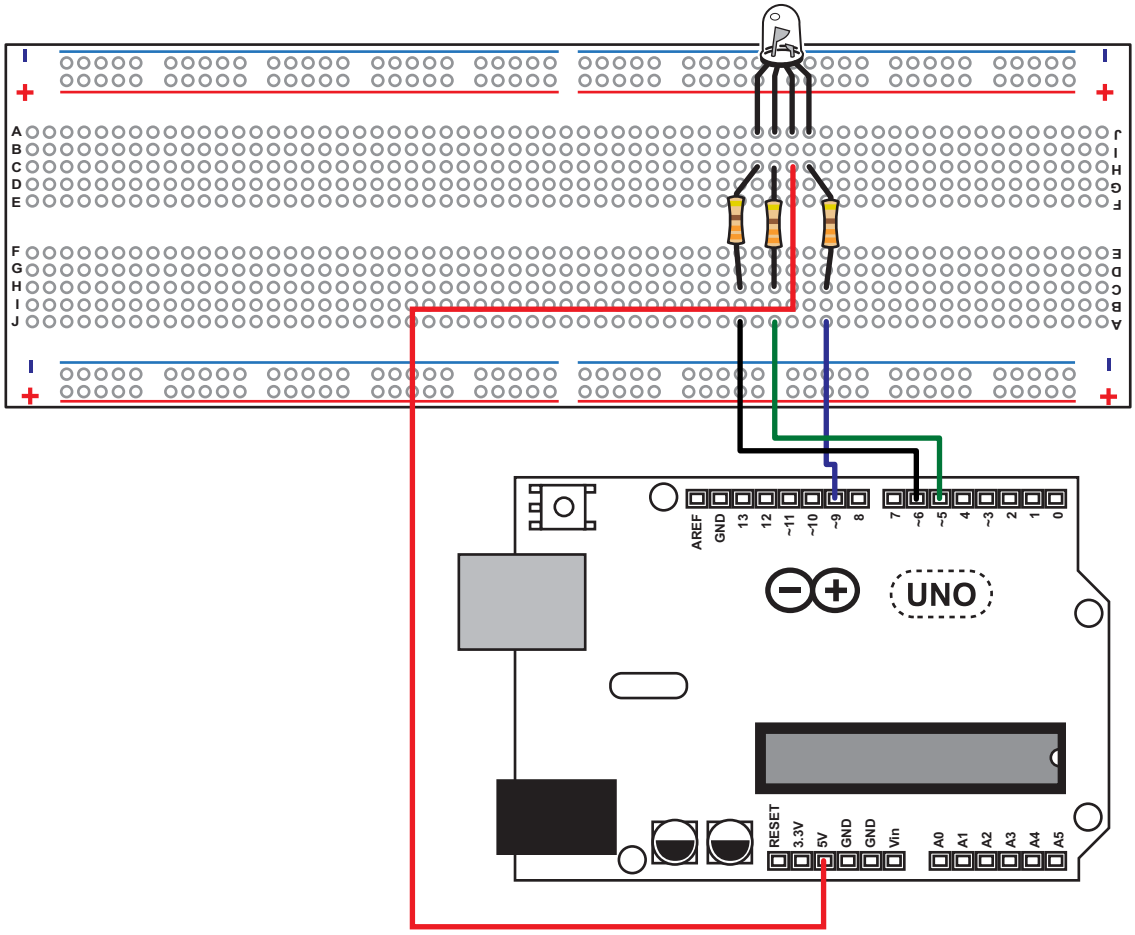
Un LED RGB permite visualizar una luz de diferente color en función de la alimentación de cada una de sus patas.

En esta práctica vamos a visualizar los colores rojo, verde y azul, alimentando para cada color una sola parte gracias al uso de pins PWM de la placa arduino.

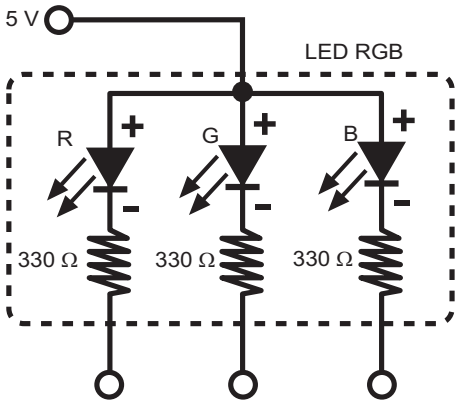
Lista de materiales con placa board

- 1 Tarjeta arduino LOG 4031
- 1 Cable USB LOG 4009
- 1 Protoboard LOG 887
- 3 reistencias de 330 Ohm (naranja-naranja-marrón) LOG 748 330
- 1 Led RGB LOG 730
- 4 Latiguillos board macho-macho 9519

Montaje en placa board



Esquema del circuito



Esquema del programa

Inicio de programa
    Bucle infinito
        Activa la salida 6 a 255
        Activa la salida 9 a 0
        Activa la salida 5 a 0
        Activa la salida 6 a 0
        Activa la salida 9 a 255
        Activa la salida 5 a 0
        Activa la salida 6 a 0
        Activa la salida 9 a 0
Activa la salida 10 a 255

Programa con Scratch

Programa con processing

```
int LED1 = 6;
int LED2 = 9;
int LED3 = 5;
int i=0;

void setup() {
        pinMode(LED1, OUTPUT);
        pinMode(LED2, OUTPUT);
        pinMode(LED3, OUTPUT);
 }

void loop() {
        analogWrite(LED1,255);
        analogWrite(LED2,0);
        analogWrite(LED3,0);

        analogWrite(LED1,0);
        analogWrite(LED2,255);
        analogWrite(LED3,0);

        analogWrite(LED1,0);
        analogWrite(LED2,0);
        analogWrite(LED3,255);
}
```

5.11. ARCOIRIS DE LUZ

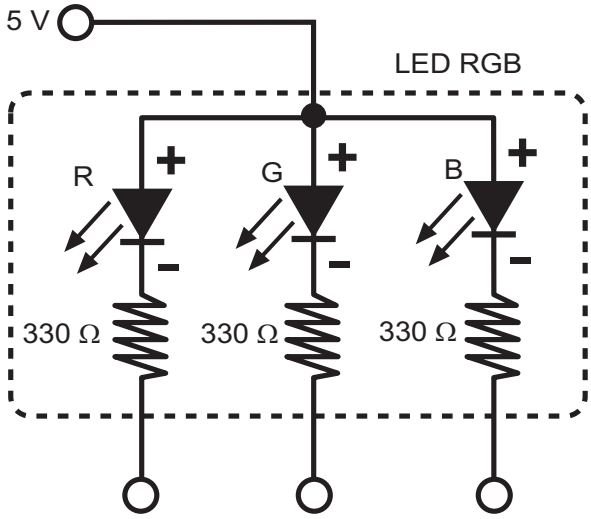
Introducción

En esta práctica el LED nos mostrara toda la gama de colores que sus características le permitan. Para ello empleamos los pines PWM de la placa Arduino y realizamos diferentes combinaciones de tensión en cada una de las patas del LED RGB.

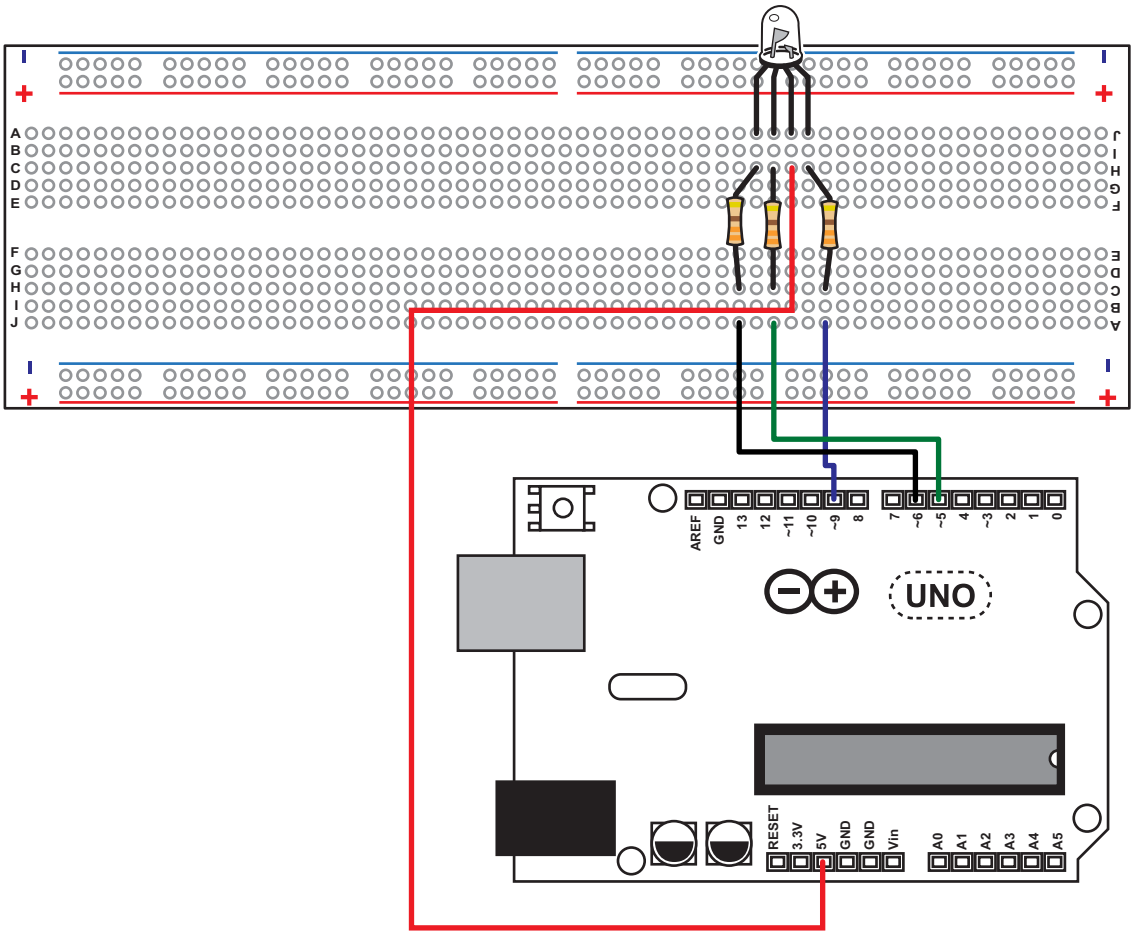
Lista de materiales con placa board

- 1 Tarjeta arduino LOG 4031
- 1 Cable USB LOG 4009
- 1 Protoboard LOG 887
- 3 reistencias de 330 Ohm (naranja-naranja-marrón) LOG 748 330
- 1 Led RGB LOG 730
- 4 Latiguillos board macho-macho 9519

Esquema del circuito



Montaje en placa board



Esquema del programa

Inicio del programa
    Bucle infinito
        Salida 5 varía de 0 a 255
        Salida 9 varía de 255 a 0
        Salida 9 varia de 0 a 255
        Salida 6 varía de 255 a 0
        Salida 9 varia de 255 a 0
        Salida 5 varia de 255 a 0
        Salida 5 varía de 0 a 255
        Salida 9 varia de 0 a 255
        Salida 5 varía de 255 a 0
        Salida 6 varia de 0 a 255
        Salida 9 varia de 255 a 0
        Salida 9 varia de 0 a 255
    Fin del bucle
Fin

Programa con Scratch

al presionar
por siempre
fijar I a 0
analógico 6 valor 255
analógico 9 valor 255
analógico 5 valor 0
repetir hasta que I = 255
analógico 5 valor I
analógico 9 valor 255 - I
esperar 0.2 segundos
sumar 1 a I
fijar I a 0
repetir hasta que I = 255
analógico 9 valor I
analógico 6 valor 255 - I
esperar 0.2 segundos
sumar 1 a I
fijar I a 0
repetir hasta que I = 255
analógico 9 valor 255 - I
analógico 5 valor 255 - I
esperar 0.2 segundos
sumar 1 a I
fijar I a 0
repetir hasta que I = 255
analógico 5 valor I
esperar 0.2 segundos
sumar 1 a I
fijar I a 0
repetir hasta que I = 255
analógico 9 valor I
analógico 5 valor 255 - I
esperar 0.2 segundos
sumar 1 a I
fijar I a 0
repetir hasta que I = 255
analógico 6 valor I
analógico 9 valor 255 - I
esperar 0.2 segundos
sumar 1 a I
fijar I a 0
repetir hasta que I = 255
analógico 9 valor I
esperar 0.2 segundos
sumar 1 a I
fijar I a 0

Programa con processing

```
int LED1 = 6;
int LED2 = 9;
int LED3 = 5;
int i=0;

void setup() {
        pinMode(LED1, OUTPUT);
        pinMode(LED2, OUTPUT);
        pinMode(LED3, OUTPUT);
}

void loop() {
        analogWrite(LED1,255);
        analogWrite(LED2,255);
        analogWrite(LED3,0);
        i=0;
        for(i=0; i<=255;i++){
                analogWrite(LED3, i);
                analogWrite(LED2,255-i);
                delay(10);    }
        i=0;
         for(i=0; i<=255;i++){
                analogWrite(LED2, i);
                analogWrite(LED1,255-i);
                delay(10);    }
        i=0;
        for(i=0; i<=255;i++){
                analogWrite(LED2,255-i);
                analogWrite(LED3,255-i);
                delay(10);    }
        i=0;
        for(i=0; i<=255;i++){
                analogWrite(LED3,i);
                delay(10);    }
        i=0;
        for(i=0; i<=255;i++){
                analogWrite(LED2,i);
                analogWrite(LED3, 255-i);
                delay(10);    }
         i=0;
        for(i=0; i<=255;i++){
                analogWrite(LED1,i);
                analogWrite(LED2, 255-i);
                delay(10);    }
        i=0;
        for(i=0; i<=255;i++){
                analogWrite(LED2,i);
                delay(10);    }
}
```

5.12. DETECTOR DE OBJETOS (CNY70)

Introducción

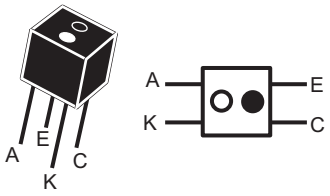
Utilizamos la capacidad del octoacoplador CNY70 para diferenciar cambios de luz reflejada, realizamos una práctica donde el octoacoplador funciona como detector de presencia. Si detecta un objeto frente a él activa un LED.

Lista de materiales con placa board

- 1 Tarjeta arduino LOG 4031
- 1 Cable USB LOG 4009
- 1 Protoboard LOG 887
- 1 reistencias de 220 Ohm (rojo-rojo-marrón) LOG 748 220
- 1 reistencias de 47K Ohm (amarillo - morado - naranja) LOG 748 47K
- 1 Led verde LOG 722
- 1 Octoacoplador CNY70 LOG 837
- 4 Latiguillos board macho-macho 9519

Esquema del circuito

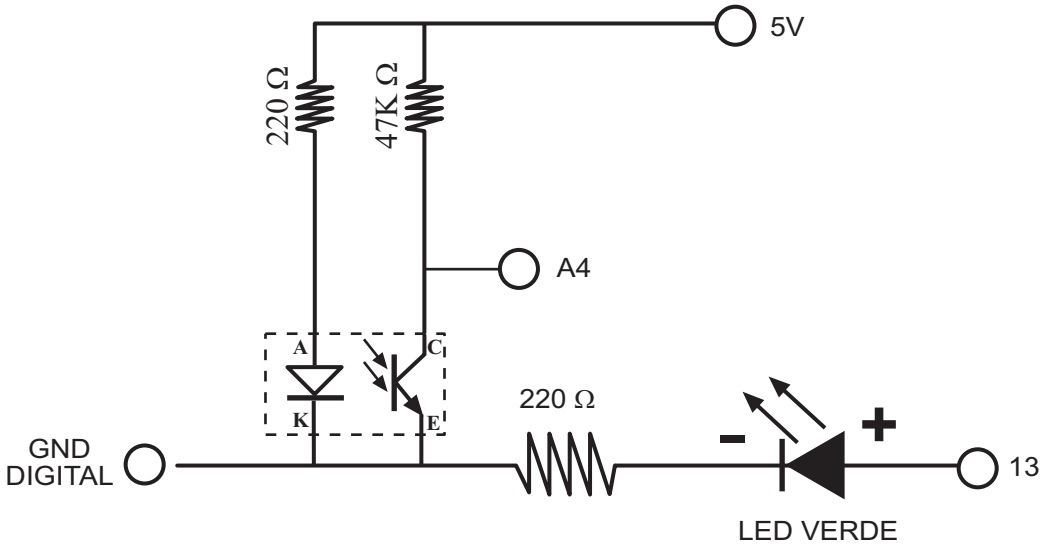
OPTOACOPLADOR CNY70 LOG 837



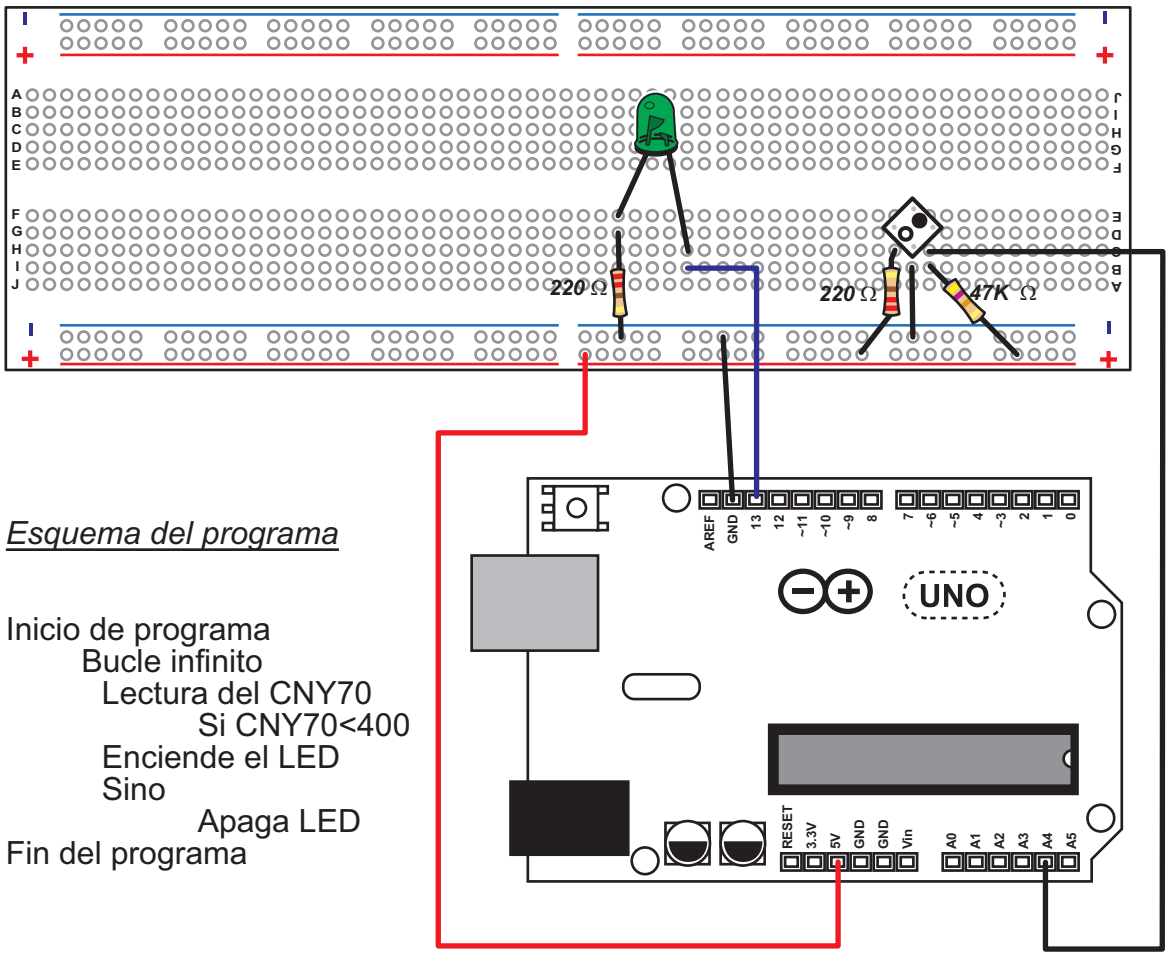
La pata "K" y "E" de los CNY70 se conectan en la misma linea.

RESISTENCIAS

220 Ω: Rojo - Rojo - Marrón
47K Ω: Amarillo - Morado - Naranja



Montaje en placa board



Esquema del programa

Inicio de programa
    Bucle infinito
        Lectura del CNY70
            Si CNY70<400
        Enciende el LED
        Sino
            Apaga LED
Fin del programa

Programa con processing

```
int LED = 13;
int sensorValue=0; //El sensor CNY70 LOG 837

void setup() {
        pinMode(LED, OUTPUT);    //Inicializacion de pin LED LOG 722 de salida
        Serial.begin(9600);
}

void loop() {

        sensorValue = analogRead(A5);   //Leemos el  sensor CNY70 LOG 837
        Serial.println(sensorValue);    // Imprimimos en puerto serie para depurar
        if (analogRead(A5)<400){     //si sensor LOG 837 destecta objeto
                digitalWrite(LED, HIGH); // activamos el LED verde LOG 722
         }
        else
        {
                digitalWrite(LED, LOW); // desctivamos el LED verde LOG 722
         }
}
```

5.13 MOVIMIENTO DE UN SERVO 180º

Introducción

Un servomotor es un motor de precisión cuyo movimiento se gstiona indicando angulos de posicionamiento.

En esta práctica el usuario introduce un valor de 0 a 180 y el servo se posiciona en ese ángulo.

Lista de materiales con placa board

- 1 Tarjeta arduino LOG 4031
- 1 Cable USB LOG 4009
- 1 Miniservo LOG 06
- 3 Latiguillos board macho-macho LOG 9519

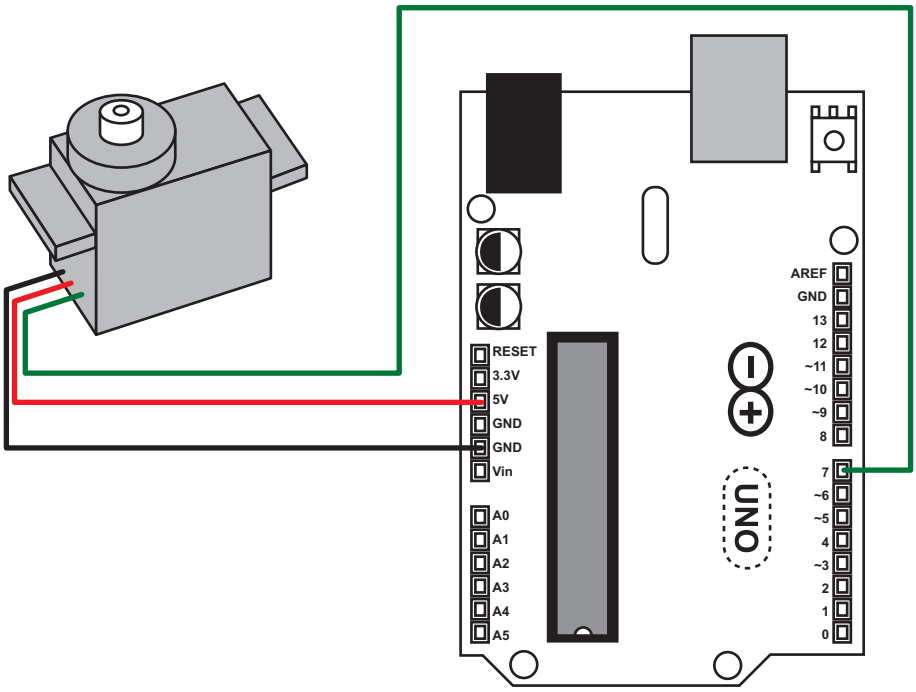
Esquema del circuito

- Cable naranja – salida 7
- Cable marrón – GND
- Cable rojo – 5V

Esquema del programa

```
Por siempre
        Pedir ángulo
        Si ángulo >= 0 y <= 180
                Posicionar motor en ese ángulo
Fin
```

Montaje en placa board



Programa con Scratch

al presionar
por siempre
preguntar Introduce el ángulo de poscionamiento (0-180) y esperar
si respuesta > 0 y respuesta < 180 o respuesta = 0 o respuesta = 180
motor 7 ángulo respuesta

Programa con processing

```
#include <servo.h>
int input;

void setup() {
      Serial.begin(9600);
      myservo.attach(7);
}

void loop() {
       if (Serial.available()>0){
            input=Serial.parseInt();
       if (input>0 && input<181){
            myservo.write(input);
            delay(15)
            }
        }
}
```

5.16. MOVIMIENTO DE UN SERVO CON UN POTENCIÓMETRO

Introducción

Conectar un servomotor a la tarjeta Arduino UNO y realizar un programa para controlar el giro del motor con un potenciómetro

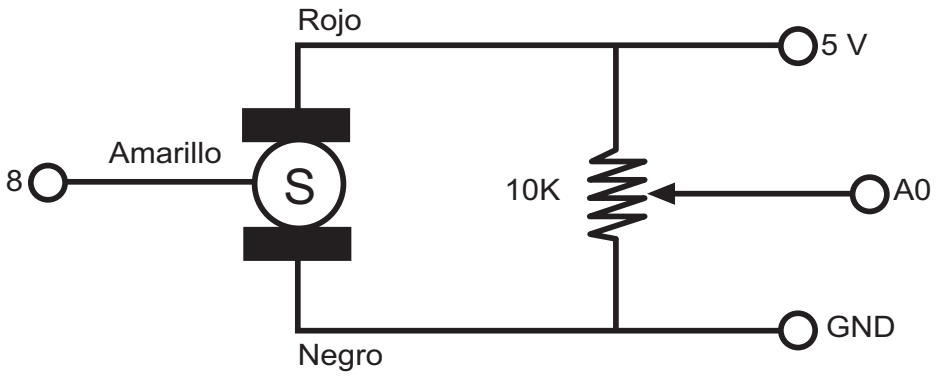
Conectamos el potenciómetro a la entrada analógica 0 y el servomotor al pin digital 8.

Introducimos en el editor de programas arduino el código indicado en el apartado 6 y cargamos el programa en la tarjeta Arduino UNO
Con un destornillador plano giramos el potenciómetro y observamos cómo el motor gira a la vez que variamos el potenciómetro y se alterna el sentido de giro en función del sentido del destornillador.

Lista de materiales

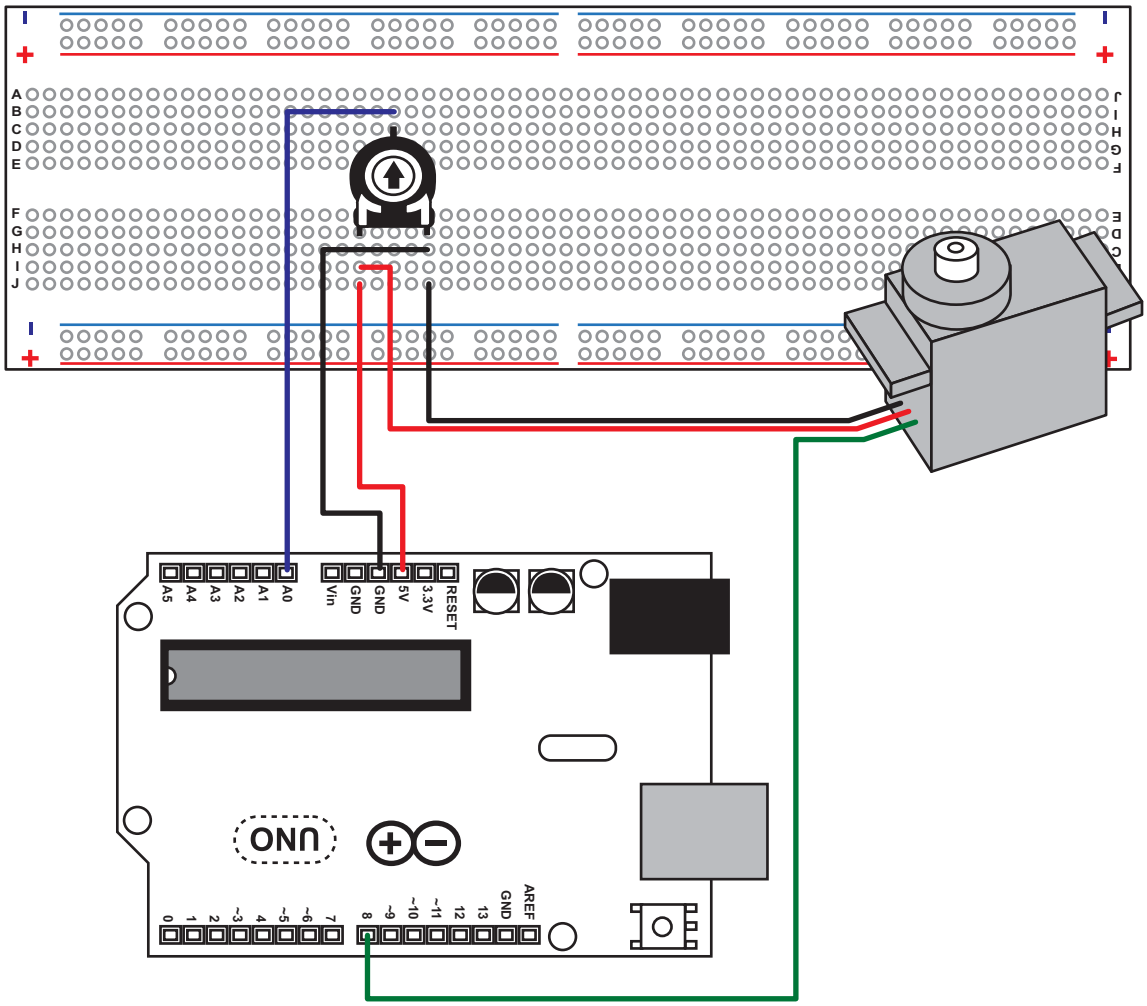
- 1 Tarjeta arduino LOG 4031
- 1 Cable USB LOG 4009
- 1 Protoboard LOG 887
- 1 Potenciómetro 10K LOG 742
- 1 Servo motor LOG 06
- 6 Latiguillos LOG 9519

Esquema del circuito



Montaje en placa board

Realizamos las siguientes conexiones:
- Conectamos la pata central del potenciómetro a la entrada analógica A0.
- Conectamos las patas traseras a 5V y GND de la tarjeta Arduino.
- Conectamos el cable rojo del motor a 5V de la tarjeta Arduino.
- Conectamos el cable marrón del motor a GND de la tarjeta Arduino.
- Conectamos el cable amarillo del motor al pin digital 8 de la tarjeta Arduino.
- Conectamos la tarjeta al puerto USB del ordenador.



Esquema del programa

Inicio de programa
    Bucle infinito
      Lectura del potenciometro
      Posicinamiento motor
      espera
Fin del progrma

Programa con processing

```
#include <Servo.h>

Servo myservo;
int potpin=0;
int val;

void setup()
{
        myservo.attach(8);
}

void loop()
{
        val=analogRead(potpin);
        val=map(val, 0, 1023, 0, 179);
        myservor.write(val);
        delay(15);
}
```

Programa con Scratch

al presionar
por siempre
motor 8 ángulo redondear valor del sensor Analog0 * 180 / 1023
decir redondear valor del sensor Analog0 * 180 / 1023 por 2 segundos

Pruebas

Al inicio el motor estará parado. Con un destornillador giramos el regulador del potenciómetro hacia la derecha. Observamos que mientras giramos el potenciómetro el motor se mueve.

Cuando dejamos de girar el regulador del potenciómetro el motor se para.

Con el destornillador giramos el regulador en sentido contrario y observamos que el motor cambia el sentido de giro y se activa mientras giramos el regulador.

Cuando dejamos de girar el regulador del potenciómetro el motor se para.

5.15. MEDIDOR DE DISTANCIAS

Introducción

Un sensor ultrasónico permite medir la distancia al objeto más cercano.

En esta práctica realizaremos un programa para medir estas distancias.

Lista de materiales con placa board

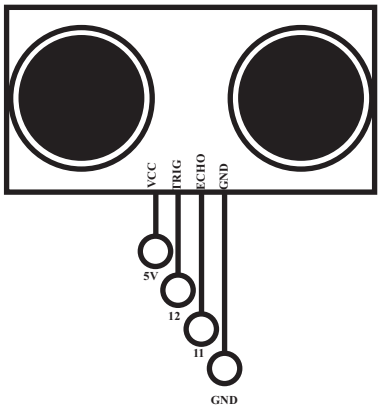
- 1 Tarjeta arduino LOG 4031
- 1 Cable USB LOG 4009
- 1 Protoboard LOG 887
- 1 Sensor ultrasónico LOG 4046
- 4 Latiguillos LOG 9519

Funcionamiento

Mide la distancia, enviando un pulso sonoro, que rebota en una superficie y capta cuanto tiempo ha tardado en rebotar en la superficie.

Conexión del ultrasonido

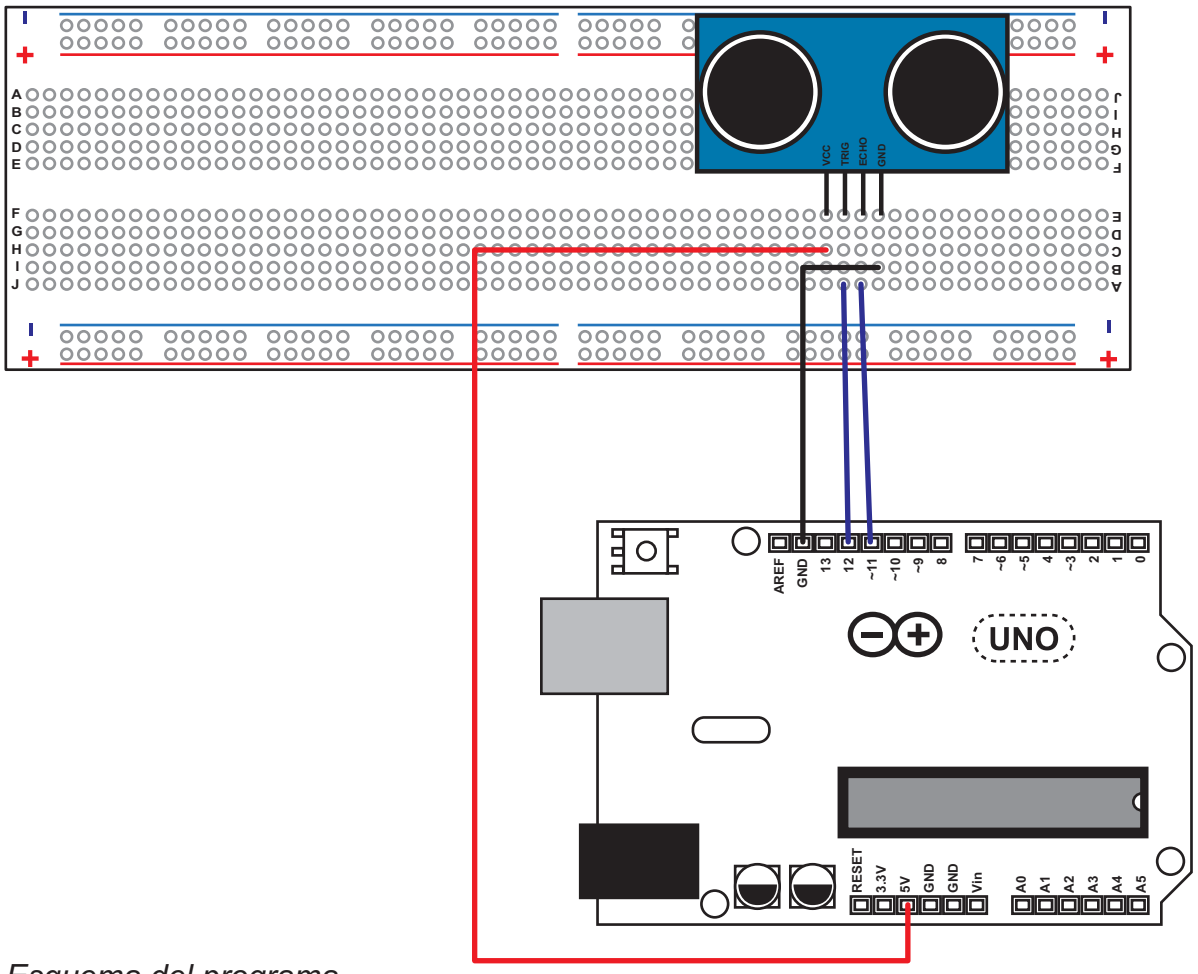
- Vcc al pin de 5 V
- Trig al pin 10
- Echo al pin 5
- GND al pin GND



Montaje en placa board

Copiar la carpeta NewPing de "entrenador arduino.zip", en la carpeta libraries que esta dentro de la carpeta de Arduino.

Realizamos las siguientes conexiones:
- Conectamos "Vcc" a la salida 5 V de Arduino.
- Conectamos "Trig" al pin digital 12.
- Conectamos "Echo" al pin digital 11.
- Conectamos "Gnd" al pin GND de la tarjeta Arduino.



Esquema del programa

Inicio de programa
    Bucle infinito
      Envio pulso de luz
      Lectura de la distancia
      Visualización distancia
Fin programa

Programa con Scratch

Para trabajar con scratch, el sensor ultrasonido lleva asociado un firmware propio (S4Afirmware14_distancia.ino). Este firmware funciona con la versión de S4A 1.5. ambos archivos se incluyen en el archivo descargable . En este firmware se han asignado los pines 10 y A5 al sensor ultrasonido, de tal forma que el pin "trig" se conectará al pin 10 y el pin "Echo" al A5. Cuando realizamos nuestro programa con S4A, la lectura del pin A5 se corresponderá con la distancia al objeto más cercano detectable.

Realizamos las siguientes conexiones:
- Conectamos "Vcc" a la salida 5 V de Arduino.
- Conectamos "Trig" al pin digital 10.
- Conectamos "Echo" al pin digital A5.
- Conectamos "Gnd" al pin GND de la tarjeta Arduino.

Programa con processing

```
#include <NewPing.h>

#define TRIGGER_PIN  12
#define ECHO_PIN     11
#define MAX_DISTANCE 200

NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE);

void setup() {
    Serial.begin(115200); // Cambiar la velocidad de lectura de monitor serie a 115200
}

void loop() {
    delay(50);
    unsigned int uS = sonar.ping();
    Serial.print("Ping: ");
    Serial.print(uS / US_ROUNDTRIP_CM);
    Serial.println("cm");
}
```

Pruebas

Abrir el "Serial monitor" de Arduino, comprobar que la velocidad esta 115200 baudio. Acercar o alejar un objeto para que nos muestre la distancia a la que se encuentra.

5.16. CONEXIÓN DISPLAY LCD

Introducción

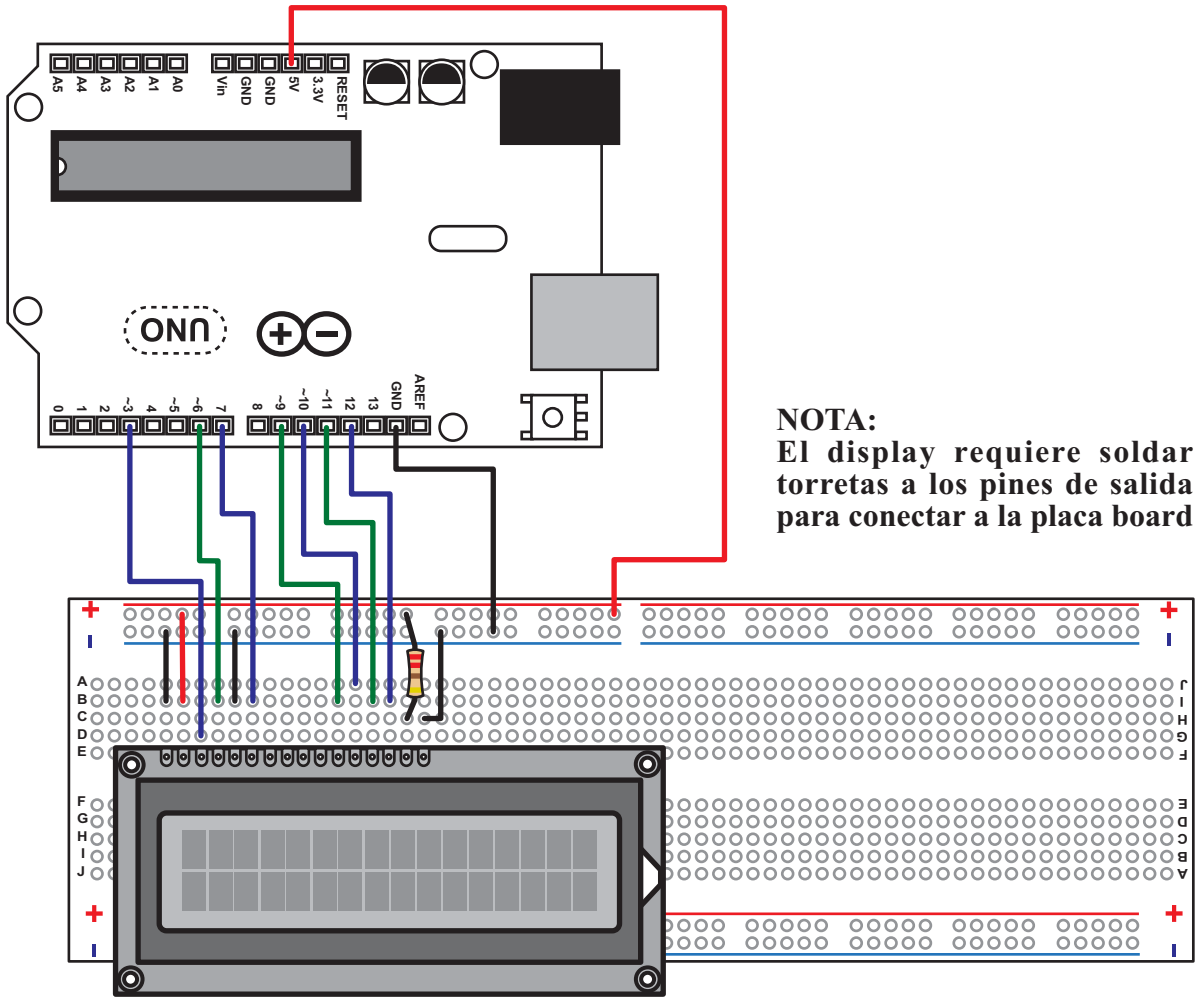
Visualización de un texto en un display LCD

Lista de materiales con placa board

- 1 Tarjeta arduino LOG 4031
- 1 Cable USB LOG 4009
- 1 Protoboard LOG 887
- 1 Display LCD LOG 4069
- 1 Resistencias de 220 ohmios (rojo-rojo-marrón) LOG 748 220
- 13 Latiguillos LOG 9519

Montaje en placa board

Realizamos las siguientes conexiones:
- Conectamos la salida 5 V de Arduino a la linea "+" (positiva).
- Conectamos un pin GND de Arduino a la linea "-" (negativa).



NOTA:
El display requiere soldar torretas a los pines de salida para conectar a la placa board

Programa con processing

```
#include <LiquidCrystal.h>

LiquidCrystal lcd(6, 7, 9, 10, 11, 12);


void setup() {
        lcd.begin(16, 2);
       analogWrite(3,100);
}

void loop() {
       lcd.print("M");
       delay(1000);
       lcd.print("I");
       delay(1000);
       lcd.print("C");
       delay(1000);
       lcd.print("R");
       delay(1000);
       lcd.print("O");
       delay(1000);
       lcd.print("L");
       delay(1000);
       lcd.print("O");
       delay(1000);
       lcd.print("G");
       delay(1000);
       for(int i=0; i<=3; i++){
              lcd.clear();
              delay(1000);
              lcd.print("MICROLOG");
              delay(1000);
       }
       lcd.clear();
}
```

5.17. CONTROL DE LEDS CON MANDO A DISTANCIA

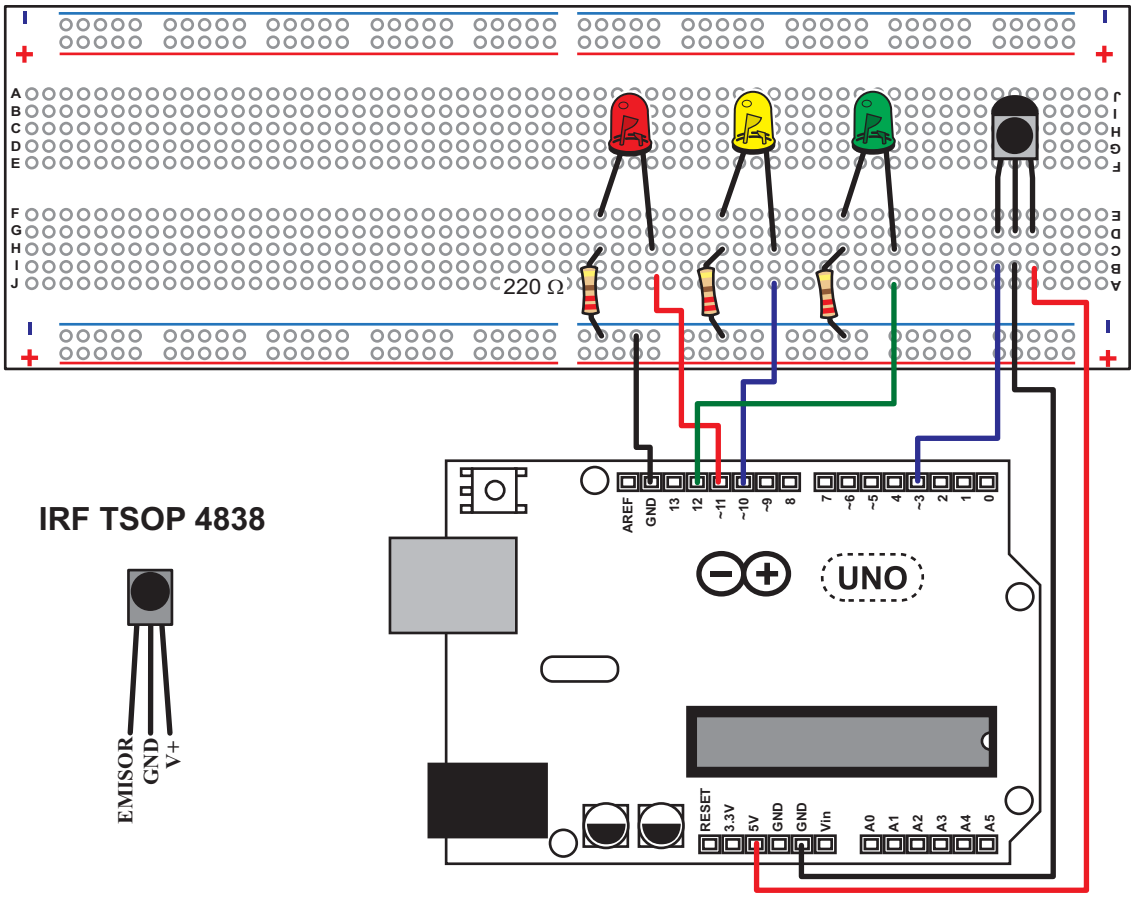
Introducción

Con ayuda de un mando a distancia por infrarrojos LOG 4000 cuya señal vamos a recibir con un TSOP LOG 835, la placa arduino encenderá y apagará 3 leds en función del botón del mando pulsado por el usuario.

Lista de materiales con placa board

- 1 Tarjeta arduino LOG 4031
- 1 Cable USB LOG 4009
- 1 Protoboard LOG 887
- 1 Led rojo LOG 724
- 1 Led amarillo LOG 723
- 1 Led verde LOG 722
- 3 Resistencias de 220 ohmios (rojo-rojo-marrón) LOG 748 220
- 1 Sensor TSOP LOG 835
- 1 Mando a distancia LOG 4000
- 4 Latiguillos board macho-macho LOG 9519

Montaje en placa board



Programa con processing

```
#include <IRremote.h>

int RECV_PIN2 = 3;
int led_verde = 12;
int led_rojo = 11;
int led_amarillo = 10;
int e_verde=0;
int e_rojo=0;
int e_amarillo=0;

IRrecv irrecv(RECV_PIN2);

decode_results results;

void setup()
{
        Serial.begin(9600);
        irrecv.enableIRIn();                    // activamos el sensor TSOP LOG 835
        pinMode(led_verde, OUTPUT);             //definimos los pines para los leds como salidas
         pinMode(led_rojo, OUTPUT);
        pinMode(led_amarillo, OUTPUT);
}

void loop() {
 if (irrecv.decode(&results)) {
        Serial.println(results.value, HEX);
         int value = results.value;
        Serial.println(value);
        Serial.println("leyendo");
        irrecv.resume();                        // Recibimos un dato del mando LOG 4000

        if (value==2295){                       //si el dato recibido es 2295, corresponde a la flecha avance - enciendo led verde
         if(e_verde==0){                        //enciendo led verde
                digitalWrite(led_verde, HIGH);
                e_verde=1;
                Serial.println("enciendo led verde");
        }
        else
        {
                Serial.println("apago led verde");
                digitalWrite(led_verde, LOW);
                e_verde=0;      }
    }
        if (value==20655){ //si el dato recibido por el LOG 835 es 20655
                if(e_rojo==0){ //enciendo led rojo
                digitalWrite(led_rojo, HIGH);
                e_rojo=1;
                Serial.println("enciendo led rojo");
                }
                else
                {
                        Serial.println("apago led rojo");
                        digitalWrite(led_rojo, LOW);
                        e_rojo=0;       }
                }
        if (value==-2041){              //si el dato recibido por el LOG 835 es 20655
                if(e_amarillo==0){ //enciendo led amarillo
                        digitalWrite(led_amarillo, HIGH);
                        e_amarillo=1;
                        Serial.println("enciendo led amarillo");
                }
                else
                {
                        Serial.println("apago led amarillo");
                        digitalWrite(led_amarillo, LOW);
                        e_amarillo=0;       }
        }

        irrecv.resume(); // recibe un dato del emisor de infrarrojo
        }

        delay(100);
}
```

5.18. MOTOR PASO A PASO

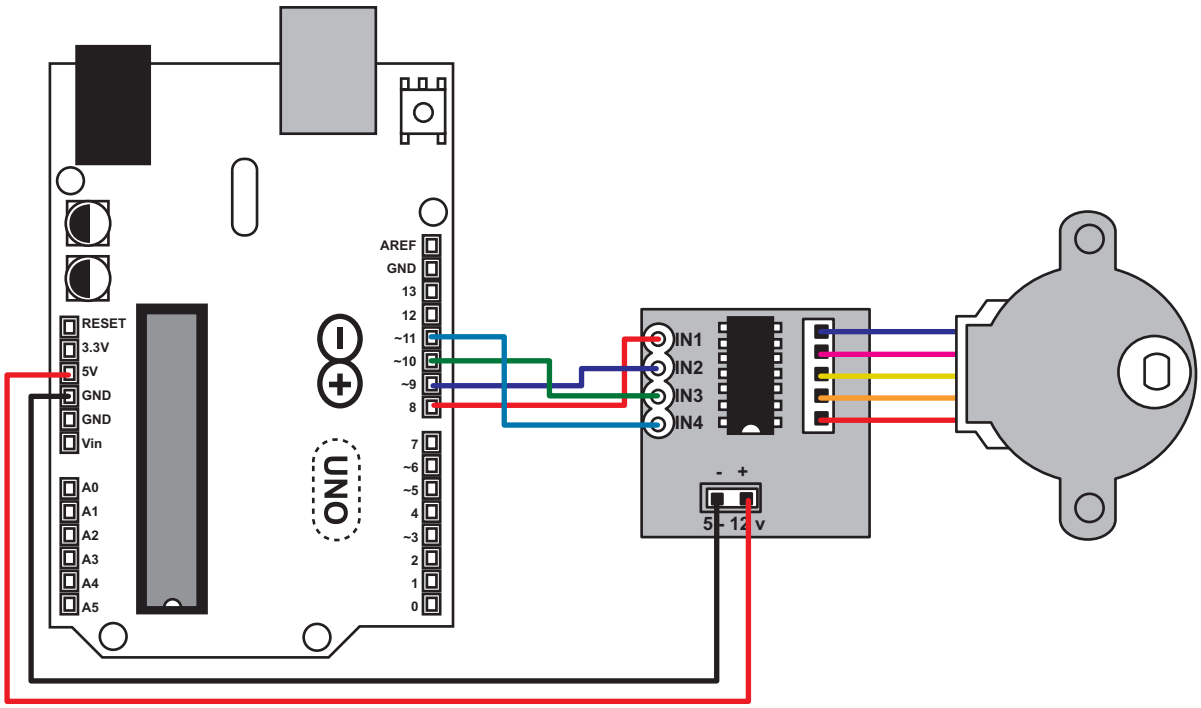
Introducción

Realizamos movimiento de avance con pausa sobre un motor paso a paso con controlador

Lista de materiales con placa board

- 1 Tarjeta arduino LOG 4031
- 1 Cable USB LOG 4009
- 1 Motor paso a paso y controlador LOG 14P
- 6 Latiguillos board macho-macho LOG 9519

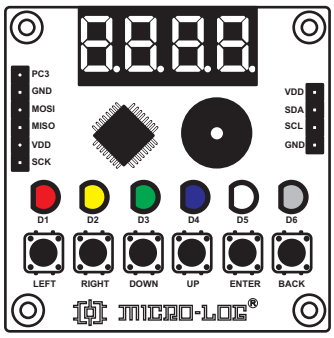
Montaje en placa board



Programa con processing

```
#include <Stepper.h>
#define STEPS 200

Stepper stepper(STEPS, 8, 9, 10, 11);

void setup(){
        stepper.setSpeed(60);
}

void loop(){
        stepper.step(100);
        delay(1000);

}
```

6. PANEL DE CONTROL PICUINO



6.1. INTRODUCCIÓN

El panel de control picuino es un shield para Arduino que se ha diseñado para realizar el control de elementos básicos de manera sencilla para alumnos principiantes. Contiene componentes que nos permiten:

- Encender y apagar 6 leds diferentes
- Activar y desactivar un zumbador
- Visualizar cifras en un dísplay de 4 díagitos con 7 segmanetos por dígito
- Control de 6 pulsadores

La placa tiene asociada una ámplia página web con información y ejercicios prácticos. Este capítulo del manual pretende ser una introducción cuya información se puede ampliar en www.picuino.com

Su facilidad de conexión, con apenas 4 cables, así como los ejemplos que se encuentran en la web facilitan tanto al profesor como al alumno el diseño y desarrollo de las clases de programación.

Su programación se realiza a través de lenguajes basados en líneas de código, pero dispone de una librería propia que facilitará el acceso y programación de cada uno de los componentes y funcionalidades.

En www.picuino.com existe toda una amplia gama de ejercicios de diferentes dificultades, para que el alumno pueda trabajar con el material desde un nivel 0 hasta alcanzar altos conocimientos y plena soltura en el manejo del lenguaje y del material.

6.2. ¿POR QUÉ UTILIZAR PICUINO?

- Reduce los errores de conexión utilizando sólo 4 cables, inequívocamente serigrafiados.

- Facilita al profesor la gestión de grandes grupos de alumnos minimizando los errores de conexión.

- Facilita al alumno el aprendizaje del proceso de programación.

- Evita la pérdida de pequeños componentes electrónicos.

- Guía web del desarrollo de clases paso a paso con unidades didácticas.

6.3. INSTALACIÓN

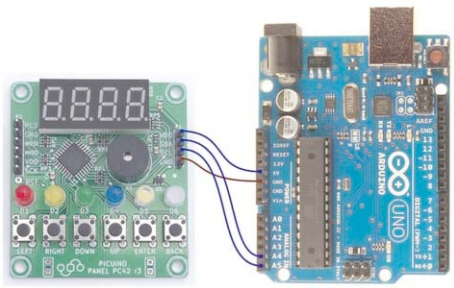
Para facilitar el proceso de programación se han elaborado librerías específicas para el shiel picuino.

Para utilizar estas librerías el primer paso es descargar el archivo desde el siguiente enlace http://www.microlog.net/PC42_r03b.zip

A continuación procedemos a instalar las librerías en el entorno Arduino IDE. Del menú Programa, seleccionamos la opción importar librería y después seleccionamos "añadir librería". Elegimos el archivo PC42_r03b.zip descargado (no es necesario descomprimirlo) y aceptamos la operación. Automáticamente la librería queda añadida a la biblioteca de arduino IDE.

6.4. CONEXIÓN DE PICUINO CON ARDUINO

Picuino se conecta a Arduino utilizando 4 cables y dejando únicamente inutilizados 2 pines analógicos:



6.5. INSTRUCCIONES BÁSICAS PROPIAS DE PICUINO

Para el control de los componentes que forman el shield de arduino, disponemos de las siguientes instrucciones:

ledWrite (nº_led, LED_OFF): Apaga el led.

ledWrite (nº_led, LED_ON): Enciende el led.

keyPressed (int *keyNum*): Devuelve 1 ó 0 en función de si el pulsador fue pulsado o no.

dispWrite (int *digit*, int *digit*, int *digit*, int *digit*): Visualiza un número en el display.

buzzFreq (int *Frequency*): Emite un sonido con una frecuencia determinada.

Puedes localizar información más ampliada en www.picuino.com

6.6 CONTROL PICUINO POR BLUETOOTH

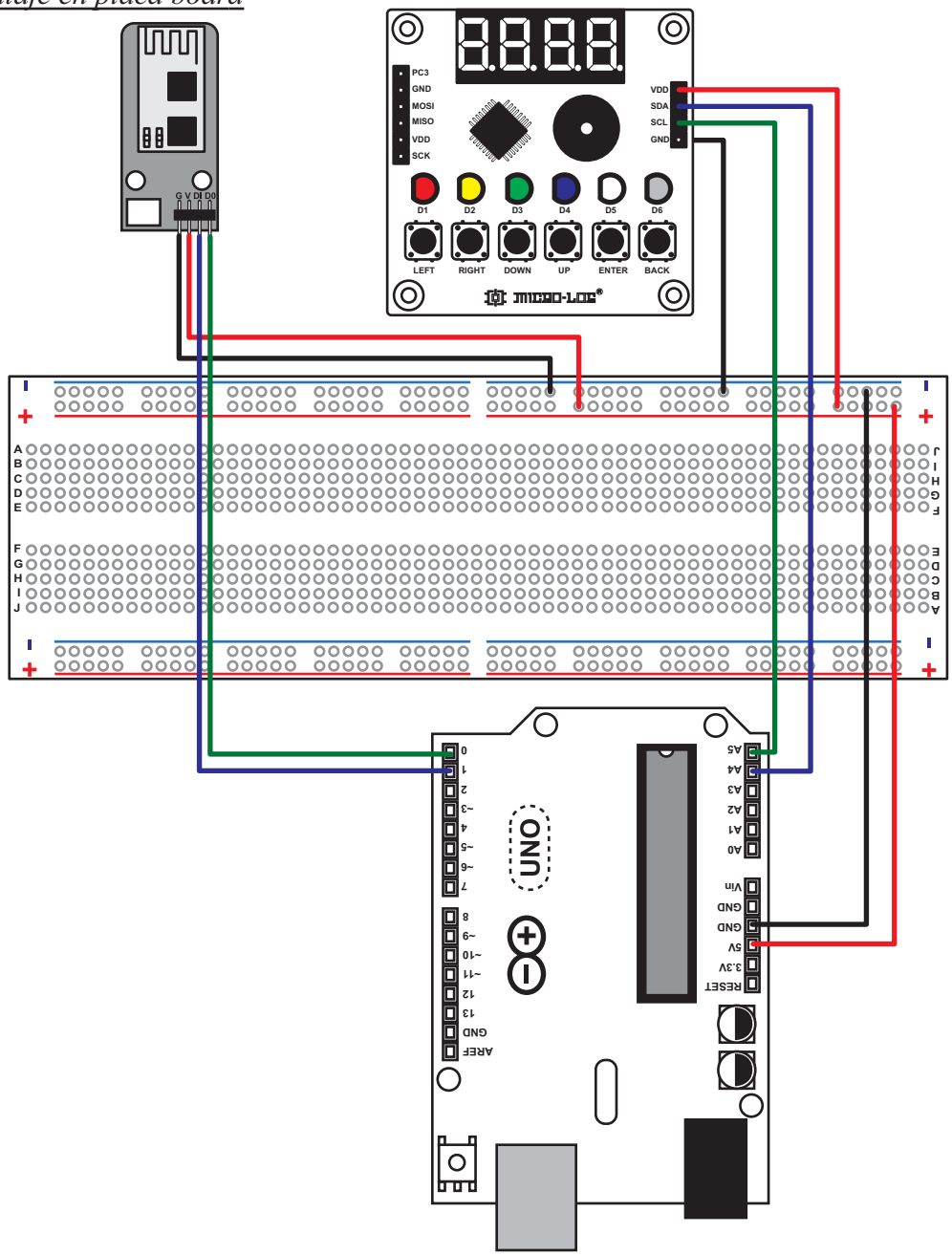
Introducción

Control de los leds rojo, amarillo, verde y azul del shield Picuino a través de una app android, gracias al uso del shield bluetooth LOG 4058.

Lista de materiales con placa board

- 1 Tarjeta arduino LOG 4031
- 1 Cable USB LOG 4009
- 1 Panel de control Picuino LOG 4055
- 1 Shield Bluetooth LOG 4058
- 10 Latiguillos board macho-hembra LOG 9518

Montaje en placa board



Programa con processing

```
#include <Wire.h>
#include <PC42.h>

int estado = 0; // almacena el dato con la orden de movimiento que envía el móvil
int rojo = 0;
int verde=0;
int amarillo=0;
int azul=0;

void setup() {
        Serial.begin(9600);
        pc.begin();
}

void loop() {
        if(Serial.available() > 0)    //activamos el puerto serie para la lectura de datos
        {
                estado = Serial.read();
        }
        Serial.println(estado);         // leemos un dato recibido a través del dispositivo HC-06 LOG 4058
        if (estado == 'r') {            // led rojo
                if (rojo==0){
                        pc.digitalWrite(1, LED_ON);
                        rojo=1;
                        Serial.println("rojo encendido");
                 }
                else{
                        pc.digitalWrite(1, LED_OFF);
                        rojo=0;
                        Serial.println("rojo apagado");
                }
        }
        if (estado == 'v') {            // led verde
                if (verde==0){
                        pc.digitalWrite(3, LED_ON);
                        verde=1;
                        Serial.println("verde encendido");
                }
                else{
                        pc.digitalWrite(3, LED_OFF);
                        verde=0;
                        Serial.println("verde apagado");
                }
        }
        if (estado == 'a'){             // led amarillo
                if (amarillo==0){
                        pc.digitalWrite(2, LED_ON);
                        amarillo=1;
                        Serial.println("amarillo encendido");
                }
                else{
                        pc.digitalWrite(2, LED_OFF);
                        amarillo=0;
                        Serial.println("amarillo apagado");
                }
        }
        if (estado == 'l'){             // led azul
                if (azul==0){
                        pc.digitalWrite(4, LED_ON);
                        azul=1;
                        Serial.println("azul encendido");
                }
                else{
                        pc.digitalWrite(4, LED_OFF);
                        azul=0;
                        Serial.println("azul apagado");
                }
        }
        delay(100);
}
```

Aplicación móvil

No está permitida la reproducción total o parcial de este libro, ni su tratamiento informático, ni la transmisión de ninguna forma o por cualquier medio, ya sea electrónico, mecánico, por fotocopia, por registro u otros métodos, sin el permiso previo y por escrito de los titulares.

MICRO-LOG®

MICRO-LOG TECNOLOGÍA Y SISTEMAS, S.L.

Andrés Obispo, 37 • 28043 Madrid
Telf. 91 759 65 22 • Fax 91 759 54 80
pedidos@microlog.es
www.microlog.es